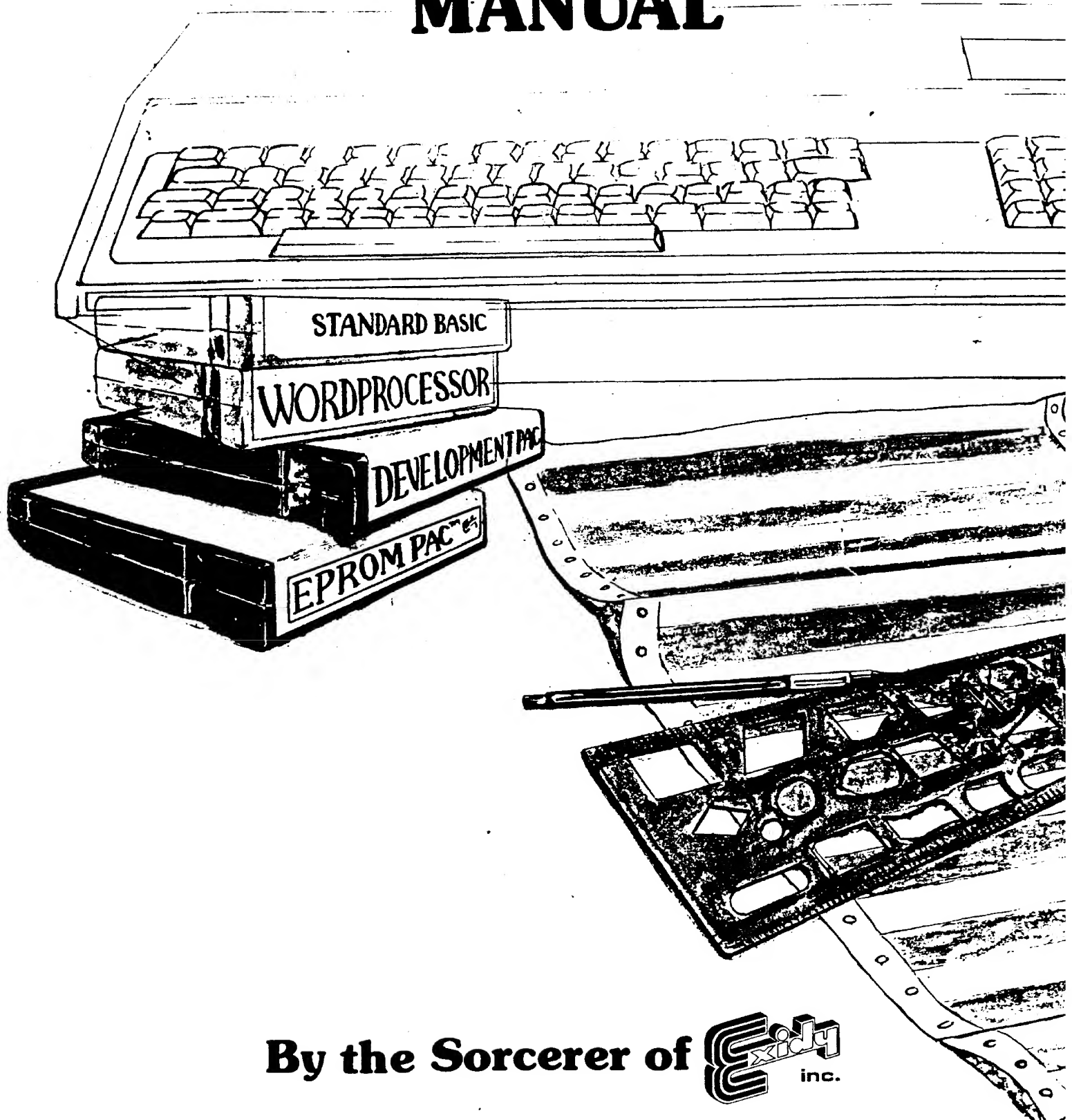


# SORCERER SOFTWARE MANUAL



By the Sorcerer of **Exidy** inc.

# SORCERER SOFTWARE MANUAL

## TABLE OF CONTENTS

<b>PREFACE</b> .....	3
<b>INTRODUCTION TO THE Z80 MICROCOMPUTER</b> .....	4
Introduction to the Z80 .....	5
Hex, Binary, and Decimal .....	5
Bits, Bytes, Addresses and "K" .....	5
RAM versus ROM .....	6
Static versus Dynamic .....	6
Z80 Architecture .....	6
<b>EXIDY SORCERER COMPUTER ARCHITECTURE</b> .....	7
Exidy Devices and Ports .....	8
Exidy Serial Port .....	8
Exidy Parallel Port .....	8
Cassette Tape File Format .....	9
Tips on Loading and Saving Files on Tape .....	9
Cassette Tape Error Checking .....	9
Programmable Graphics Character Set .....	10
Exidy Keyboard Architecture .....	11
Performing Keyboard Input .....	12
Cursor Positioning .....	12
<b>EXIDY STANDARD BASIC</b> .....	14
BASIC Floating Point Format .....	15
BASIC Control Area .....	16
Format of BASIC String Variables and Arrays .....	17
Format of BASIC Program Statements .....	17
Format of BASIC Floating Point Variables and Arrays .....	17
BASIC to Z80 Assembly Language Interface .....	18
<b>EXIDY POWER-ON MONITOR</b> .....	19
Monitor Workarea .....	20
Exidy Monitor Memory Map .....	20
Monitor Subroutines .....	22
<b>MONITOR LISTINGS</b> .....	23
<b>SUMMARY</b> .....	64

## SORCERER SOFTWARE MANUAL

### TABLES

Table 1	Z80 Registers .....	6
Table 2	Sorcerer I/O Port Assignment .....	8
Table 3	Cassette Tape File Format .....	9
Table 4	Character Codes .....	10
Table 5	BASIC Control Area .....	16
Table 6	Monitor Memory Map .....	20
Table 7	Monitor Workarea .....	21
Table 8.	Monitor Subroutines .....	22

# INTRODUCTION TO THE Z80

Before you can understand how the Exidy really works, a few fundamentals have to be covered about the architecture of the Z80 MPU (MicroProcessing Unit). First of all, let's discuss the concept of "hex."

## Hex, Binary, and Decimal

"Hex" is short for hexadecimal. This is a number system based on sixteen, not 10 as we are used to (decimal). In decimal, we have ten possible digits, 0, 1, 2, . . . , 8, and 9. In hex, we have sixteen. Of course the first ten are 0 through 9 as with decimal. But there are six more, A, B, C, D, E, and F. "A" means 10, "B" means 11, "C" 12, "D" 13, "E" 14, and "F" 15. So a number like 1CB3 makes sense in hex. In decimal numbers each digit represents a "power" of 10, namely "ones," "tens," "hundreds," and "thousands." For example, the decimal number 1895 means 1 thousands plus 8 hundreds plus 9 tens plus 5 ones, or

$$1895 = 1 \times 1000 + 8 \times 100 + 9 \times 10 + 5 \\ = 1000 + 800 + 90 + 5$$

In hex however, each digit (0 through F) represents a power of 16, "ones," "sixteens," "two hundred fifty sixes," and "four thousand ninety sixes." For example, the hex number 1895 can be written as in the example above

$$1895 = 1 \times 4096 + 8 \times 256 + 9 \times 16 + 5 \\ = 4096 + 2048 + 144 + 5 \\ = 6293 \quad (\text{decimal})$$

Another hex number 3CF1 can be seen as

$$3CF1 = 3 \times 4096 + 12 \times 256 + 15 \times 16 + 1 \\ = 12288 + 3072 + 240 + 1 \\ = 15601 \quad (\text{decimal})$$

The reason why understanding the hex number system is so important is because the majority of computers today, big, mini, and micro, are based entirely on hex. This includes the Z80 MPU, which is the basis of the Exidy Sorcerer. Its machine language instructions are in hex; its arithmetic is done in hex; characters typed on the keyboard, displayed on the screen, placed on cassette tape and printed on a printer are all in hex.

If you understand hex, then "binary" (the number system based on 2) should present no problems. There are only 2 digits possible to make any binary number, 0 and 1. These binary digits are called "bits." A bit can be 0 or 1. Each of these digits represents a power of 2 (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, and 32768). So a number in binary like 0011110011110001 is

$$0011110011110001 = 0 \times 32768 + 0 \times 16384 + 1 \times 8192 + 1 \times 4096 + \\ 1 \times 2048 + 1 \times 1024 + 0 \times 512 + 0 \times 256 + \\ 1 \times 128 + 1 \times 64 + 1 \times 32 + 1 \times 16 + \\ 0 \times 8 + 0 \times 4 + 0 \times 2 + 1 \\ = 8192 + 4096 + 2048 + 1024 + \\ 128 + 64 + 32 + 16 + 1 \\ = 15601 \quad (\text{decimal})$$

But that means, according to the previous example, that since 15601 decimal is also 3CF1 hex, then

$$0011110011110001 \text{ (binary)} = 3CF1 \text{ (hex)}.$$

This is no mere coincidence. Let's see why. If we look at a "4-bit binary number" (i.e., a number in binary made up of only four digits of 0's and 1's), then the smallest it could be is 0000 (0 decimal), and the largest it could be is 1111 (15 decimal or F hex). Thus every digit in hex, 0-F, can be expressed exactly as a 4-bit binary number:

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

In other words, a hex digit is really just another way of writing 4 bits, or, every 4 bits of a binary number can be grouped as 1 hex digit. Let's see how that works with the numbers we just did. 001110011110001 can be broken into groups of 4 bits (right to left) as follows:

0011<sup>1</sup> 1100 1111 0001

If each 4-bit group is viewed individually, they calculate to

$$0011 = 3 \text{ decimal (3 hex)} \\ 1100 = 12 \text{ decimal (C hex)} \\ 1111 = 15 \text{ decimal (F hex)} \\ 0001 = 1 \text{ decimal (1 hex)}$$

So it can be written

0011 1100 1111 0001      binary  
3      C      F      1      hex

So hex and binary are actually the same thing, with different groupings. Another example, to write 0F8D hex in binary

0      F      8      D      hex  
0000 1111 1000 1101      binary

which, when pieced back together, becomes

$$0000111110001101 = 0F8D.$$

## Bits, Bytes, Addresses, and "K"

Enough about decimal, hex, and binary. We now know how numbers are written on the Z80. Let us take a look at how memory is organized.

The smallest unit of information that can be placed in the memory of just about any computer made, including the Z80, is a bit, the same bit we saw earlier. This only holds a 0 or a 1 however, and is too small for normal numerical use. So a larger unit was created, called a "byte." A byte is just eight bits or two hex digits grouped together.

So a byte can contain a number from 00000000 binary (00 hex, 0 decimal) to 11111111 binary (FF hex, 255 decimal). Each unique byte in the Exidy's memory space is assigned a four-hex digit (two-byte) number called an "address." This address identifies the particular byte and its contents. Addresses start at 0000 hex and end at FFFF hex (65535). Thus, the Exidy (Z80) can have up to 65536 bytes of memory. Another way programmers like to put this is to use the term "K." A "K" is just another way of saying the number 1024 decimal (400 hex). So 65536 boils down to 64K (64x1024 = 65536).

## RAM versus ROM

Since we are on the subject of memory, there are two types. In one type the contents can never be changed. Information can only be "read" from it. This is called **Read Only Memory** or **ROM** (computerists love abbreviations or acronyms). ROM is usually used to contain programs or data which is to be present in the same state all the time. For example, the Exidy Monitor program is in ROM (starting at memory byte address E000) and Exidy BASIC is in ROM (the ROM-PAC starting at address C000). ROM can have its contents "burned in" permanently at the factory, or can be burned in once by the programmer (called PROM or Programmable ROM), or can be erased by strong ultraviolet light and burned in over and over again (called EPROM or Eraseable PROM).

However, for programmers to write and run programs, we need memory which we can change or modify the contents. This is called **Random Access Memory** or **RAM**. When the size of an Exidy's memory is given (e.g., 8K, 16K, 32K), this number applies only to RAM, or user-modifiable memory. All Exidys have the same ROM area potential. So a 16K Exidy has 16x1024 or 16384 bytes of RAM.

## Static versus Dynamic

The above two terms are usually only applied to RAM. Static RAM has the ability to hold its contents indefinitely as long as electrical power is applied. Dynamic RAM, on the other hand (in milliseconds usually), loses or leaks its contents, and the data must be re-written or refreshed to the RAM often enough to keep the data from disappearing altogether. Typically static RAM requires more power, is more expensive, but is faster. The Exidy and many other Z80 based systems use dynamic RAM because of power and cost considerations, and also because the Z80 MPU is well-suited to interface to dynamic RAM (e.g., it can be made to do the RAM refreshing).

---

## Z80 ARCHITECTURE

The Z80 microprocessor is an 8-bit based machine. In other words, its data flow and arithmetic is usually on a 1-byte basis. It can address up to 64K bytes of memory. On the Exidy, a maximum of 32K bytes of this can be placed onboard (in the keyboard unit), while another 16K can be located as ROM for the Monitor and various ROM cartridges.

In addition to having 64K of possible memory, the Z80 has twenty-two registers. These are special high speed memories which reside on the MPU chip, and are used for arithmetic and program logic functions. These are all 1 byte in size unless otherwise noted:

**Table 1. Z80 Registers**

A	— the accumulator. This is the central register
F	— the flags register. Each bit represents a CPU status; e.g., the "Z" bit is on if the A register contains 0; the "S" bit is on if A is negative
B	— general use register
C	— general use register
D	— general use register
E	— general use register
H	— general use register
L	— general use register
SP	— 2-byte register containing the current stack address
PC	— 2-byte program counter containing the address of the next instruction to be executed.
IX	— 2-byte index register. Usually will contain an address to be used with a constant offset or displacement.
IY	— 2-byte index register with the same type of use as IX.
I	— register used to allow processing of external interrupts to the Z80 from the S-100 bus
R	— refresh register which can be used to provide dynamic RAM refreshing operations.

Registers A, F, B, C, D, E, H, and L have an alternate register called A', F', B', C', D', E', H', and L'. Only one set can be used at a time, while the other set allows space to save important program information. The EXX and EX Z80 instructions are used to flip back and forth between them. Also some registers can be connected together to create 2-byte, 16-bit register pairs. These are AF, BC, DE, and HL.

For more detailed information on the Z80 MPU the reader is referred to the Zilog publication "Z80 CPU, Z80A CPU Technical Manual," product number 03-0029-01.

# **EXIDY SORCERER COMPUTER ARCHITECTURE**

# Exidy Devices and Ports

The Sorcerer has the following I/O devices or ports. Listed also is the Monitor command(s) to activate each:

**Table 2. Sorcerer I/O Port Assignment**

a. the keyboard	SET I=K
b. the video screen	SET O=V
c. cassette tape #1	SET I=S, SET O=S
d. cassette tape #2	SET I=S, SET O=S
e. serial RS-232 interface	SET I=S, SET O=S
f. parallel interface	SET I=P, SET O=P
g. Centronics printer interface	SET O=L

Note that these are onboard ports. This list does not include any devices added to the Exidy via the S-100 bus expansion facility.

The keyboard is implemented as part of the Z80 I/O port number FE hex (254), input bits 0-4, output bits 0-3. The video screen needs no port but uses the 1920-byte RAM area at address F080 as a 64 by 30 screen. There is a port FE bit (input 5) indirectly related to video processing which signals when vertical retrace is in progress on the TV screen. The two cassette interfaces are part of the serial interface and provide an audio translation of the digital data suitable for recording on tape quite reliably.

## Exidy Serial Port

The serial port allows data transfer to occur between the Exidy and external devices (such as printers, modems, cassette tape, and the like). Data travels one bit at a time in a predefined conventional sequence called asynchronous transmission protocol.

The protocol defines how the data is to look, and the speeds at which it is to travel. For example, each 8-bit byte of data is actually sent as a 10- or 11-bit stream, sometimes even longer. The 8 bits must be preceded by a bit called a start bit, and must be followed by one or usually two or more stop bits. These bits also must be sent and received at a particular speed, predetermined by the sender and receiver. The speed is given in bits per second, or commonly called "baud" (derived from Baudot, the name of one of the forerunners of terminal communications). Thus, 300 baud means 300 bits per second. Since it takes about 10-11 bits to transmit a byte or character, this means about 30 characters per second. The Exidy serial interface "speaks" this common language, and operates at one of the two speeds, either 1200 baud (120 cps) or 300 baud (30 cps).

The serial port is actually two devices, an RS-232C interface and the dual cassette interface. RS-232C is the name given to a widely accepted standard of signal voltage and logic levels and the pinouts of the 25-pin plug or connector used for cabling between the sender and receiver. The asynchronous protocols signals are usually sent via this RS-232C standard. Another part of Z80 port FE (output bit 7) determines whether the serial port is RS-232C (bit on) or dual cassette (bit off). Cassette is the default. Output bit 6 controls the baud rate (1 = 1200, default, 0 = 300). Port status is placed on port FD while data transfer occurs on FC. For example, to connect a 300

or 1200 baud RS-232C serial printer to the Exidy, follow instructions given with the printer and from Exidy. However, the following guidelines may be used:

1. Connect pin 7 of the serial DB25 connector to printer ground pin 7.
2. Connect pin 3 to printer pin 2.
3. Connect pin 2 to printer pin 3.

Reset the Exidy, enter the Monitor (BYE in BASIC), enter the command SET O=S, and all output which would have gone to the screen will go to the printer, until Reset or SET O=x is entered (x is usually V to return to video). There is also software available from Exidy providing a serial driver, and the ability to use the serial interface to turn the Sorcerer into a dumb terminal connected to another computer. Typically a modem and possibly an acoustic coupler may be required here. Reverse pins 2 and 3 in the above guidelines for this use.

The cassette interfaces may also be used with motor control. Pins 12 and 24, 13 and 25 can be used to turn cassette number 1 and 2 off and on for SAVES, LOADs, FILEs and BATCHs commands. Pins 15, 5 and 20, 16, 18, and 21 are the mike input, auxiliary input, and earphone output connections. Note that cassette number 1 has these mike and ear connections duplicated as RCA plugs on the back of the Sorcerer.

## Exidy Parallel Port

The parallel port differs from the serial port mainly in that data is transferred an entire byte at a time. This is ideal for fast printers and sometimes even some floppy disk units. The Sorcerer also provides an interface to the popular Centronics printer. The same parallel port is used, but unique software "handshaking" is done by the Monitor I/O driver. An example of the handshaking which occurs between the Sorcerer and printer might be the following "electronic conversation" over port FE, the parallel interface status port:

Printer: "Wait, I'm still busy, send no data."

"OK, now you can send."

Exidy: "Here it is, let me know when I can send more."

The 8-bit (and at times status) rides on port FF.

To successfully hook up a Centronics or Centronics-like printer to the parallel port, again follow the printer's and Exidy's instructions. Here are some additional guidelines:

1. Connect parallel pins (DB25 connectors again) 5-7 and 16-19 (data bits 0-6) to the printer's data lines 0-6 (see printer's pinouts).
2. Connect pin 4 (data output bit 7) to the printer's input strobe line, a negative (true is low, false is high) pulse indicating data is ready to be transmitted.
3. Connect pin 1 to the printer ground.
4. Connect pin 25 (input data bit 7) to the printer busy line, indicating the printer is not ready to accept any data (probably still printing previous data).
5. Pins 2 and 3 (output accepted and available) and others may also be required depending on the printer model.

Once this is done, Reset the Exidy, enter the Monitor, type in the command SET O=L, and from that point on all output will be routed to the screen and the printer, until Reset occurs or until another SET O=x command is entered.

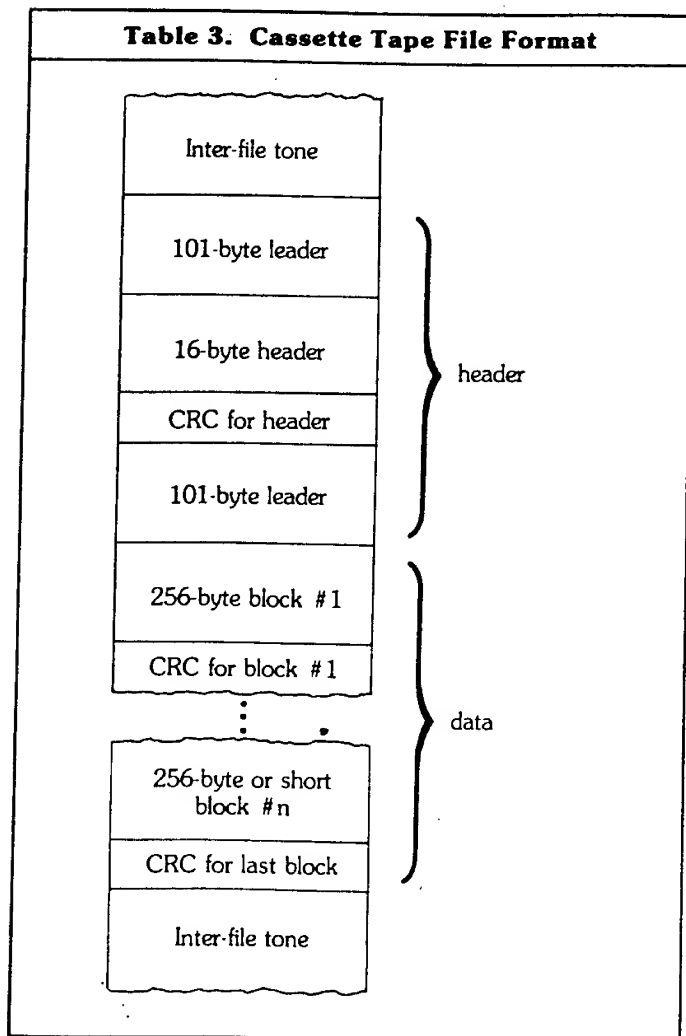
# CASSETTE TAPE FILE FORMAT

When a SAVE, LOAD, or FILES command is done from the Monitor, or when a CSAVE or CLOAD is done from BASIC, files are processed from the cassette tape device on the serial interface. This applies to both cassette #1 and #2. Cassette tape motor-on routine can be found at E024 (-8156), motor-off at E027 (-8153), cassette save at E02A (-8151), and cassette load at E02D (-8148).

Cassette files on the Exidy have the following appearance, whether at 300 or 1200 baud:

1. Inter-file tone
  - a. a high frequency tone always output by the cassette interface when data is not present.
2. 101-byte leader
  - a. 100 bytes of 00 (nulls)
  - b. 1 byte of 01 (control-A or SOH, Start-Of-Header).
3. 16-byte file header (see description in MWA above).
4. CRC for header
  - a. 1 byte CRC for error checking. Details later.
5. Up to 256 bytes of data.
6. CRC for above data block (1 byte again).
7. Repeat 5 and 6 until data exhausted. The last data block may be short (less than 256 bytes). CRC still follows.
8. Inter-file tone (same as before the file).

This format is used by both BASIC and machine language files. It is depicted pictorially as follows:



To LOAD or CLOAD a file, or to perform a FILES command, the Monitor scans the tape (whichever is on) for the leader. Then the header is read into the MWA and the "FOUND..." message is sent to the current SEND device. The data portion is then either skipped (wrong file, or FILES command) or loaded. All CRCs are always validity checked for any of these commands. Thus, to check all the bits on an entire tape for errors, it is sufficient to perform a FILES command.

Note that the default tape transfer rate is 1200 baud. A much more reliable method of saving data is to use 300 baud. However it will take four times longer to SAVE and LOAD, and use a lot more tape. This is accomplished with the SET T=1 command.

Still, even at 1200 baud, the Sorcerer tape system is the best I've come across. It is the most reliable, and with its file headers, it is the easiest to use. The user does not even need a recorder with a tape digital counter to find files with these headers. The cleverness of the tape system makes the Exidy basic offering (just cassette, no expansion to S-100 capability, diskette, etc.) a very attractive low-priced system.

## Tips on Loading and Saving Files on Tape

The following hints can be used to minimize problems with cassette recording of files:

### To Load:

1. Use a relatively inexpensive cassette recorder (\$30-\$60) with ALC (Automatic Level Control). This means you have no control over the volume or tone of the recordings. All are made exactly the same way. Strangely enough, experience shows that expensive recorders work worse.
2. Connect the MIC wire to the microphone input. Do *not* use the auxiliary input on most recorders. The signal will be too weak.
3. Connect the EAR wire to the earphone or monitor jack.

### To Play:

1. You must find the correct volume and tone for your recorder. As a first guess, set volume and tone to 7-8 out of 10, or 3/4 high.
2. Listen to the tape play through the speaker. The intra-file tone should be louder than normal listening volume, maybe even as loud as possible without distortion and noise. The data should sound high-pitched and clear, like static.
3. Try loading a file. Tinker with volume and tone until at least a file header is read without a CRC error ("FOUND..." message appears). Now you are close enough to the correct settings.
4. Once found, the correct settings should be able to be used for all tapes recorded on that recorder.

## Cassette Tape Error Checking

The CRC (Cyclic Redundancy Check) method is used to detect bit transmission errors in cassette data recordings. The CRC is stored at MWA + 46. CRC checking is done with this algorithm: When the file is first written to tape (i.e., when the 101-byte leader is written), the CRC is 0'd. For every data byte, in program or header, the current CRC is subtracted from the data (data-CRC), and the ones complement of this is used as the next CRC for the next byte (i.e., FF - (data - CRC), or all the bits are flipped - 0's become 1's, and 1's 0's). When the file or block is completely written, the current CRC is written as the final byte. Note: this is why BASIC programs grow by one byte every time they are loaded and re-saved. When the file is loaded again, the CRC is calculated again as above, and is compared to the last byte of the block (the CRC written). A match means no errors (almost always), while a mismatch means an error. This is identical in BASIC files as in machine language files, since the same Monitor routines are used to write/read tapes.

# Programmable Graphics Character Set

Each byte in memory can contain exactly one character which can be input from the keyboard, displayed on the video, printed, etc. Thus, there are 256 possible combinations of these characters (00-FF, 0-255). These codes can be mapped as follows on the Exidy. Again, codes are given in both hex and decimal.

**Table 4. Character Codes**

Code	Description	Code	Description
00-7F	0-127	128 standard ASCII characters:	
00-1F	0-31	32 ASCII control characters (e.g., CR, LF, etc.).	
20	32	ASCII blank	
21-2F	33-47	ASCII punctuation	
30-39	48-57	ASCII numbers 0-9	
3A-40	58-64	ASCII punctuation	
41-5A	65-90	ASCII upper case A-Z	
5B-60	91-96	ASCII punctuation	
61-7A	97-122	ASCII lower case a-z	
7B-7F	123-127	ASCII punctuation and "delete" character (7F)	
80-BF	128-191	64 standard Exidy keyboard graphics. These are obtained by depressing the GRAPHICS key	
CO-FF	192-255	64 programmable graphics characters. These are obtained by depressing SHIFT and GRAPHICS keys:	
C0	192	GRAPHIC SHIFT 1	
C1	193	2	
C2	194	3	
C3	195	4	
C4	196	5	
C5	197	6	
C6	198	7	
C7	199	8	
C8	200	9	
C9	201	0	
CA	202	:	
CB	203	- (hyphen)	
CC	204	<	
CD	205	(tab)	
CE	206	Q	
CF	207	W	
D0	208	E	
D1	209	R	
D2	210	T	
D3	211	Y	
D4	212	U	
D5	213	I	
D6	214	O	
D7	215	151	— S
D8	216	152	Q SPACE
D9	217	153	O PLANT
DA	218	154	P
DB	219	155	P
DC	220	156	L
DD	221	157	L
DE	222	158	A
DF	223	159	A
E0	224	160	Y
E1	225	161	F
E2	226	162	15
E3	227	163	DIAMOND
E4	228	164	CLUE
E5	229	165	B
E6	230	166	B
E7	231	167	B
E8	232	168	B
E9	233	169	B
EA	234	170	B
EB	235	171	/
EC	236	172	/
ED	237	173	+
EE	238	174	-
EF	239	175	-
F0	240	176	-
F1	241	177	-
F2	242	178	-
F3	243	179	-
F4	244	180	-
F5	245	181	-
F6	246	182	-
F7	247	183	-
F8	248	184	-
F9	249	185	-
FA	250	186	-
FB	251	187	-
FC	252	188	-
FD	253	189	-
FE	254	190	-
FF	255	191	-
			P
			[
			]
			A
			S
			D
			F
			G
			H
			J
			K
			L
			:
			@
			_ (underscore)
			Z
			X
			C
			V
			B
			N
			M
			, (comma)
			. (period)
			/ (slash)
			- (on numeric pad)
			7 (on numeric pad)
			8 (on numeric pad)
			9 (on numeric pad)
			+ (on numeric pad)
			4 (on numeric pad)
			6 (on numeric pad)
			x (on numeric pad)
			1 (on numeric pad)
			2 (on numeric pad)
			3 (on numeric pad)
			+ (on numeric pad)
			0 (on numeric pad)
			. (on numeric pad)
			= (on numeric pad)



Each of the preceeding 64 characters can be defined to be any design or shape desired. Each consists of 8 bytes in memory, or 64 bits. These sets of 8 bytes (64 of them) start at address FE00 (-512). On the screen each character consists of 8 lines of 8 dots, or 64 dots. Thus, each of the 8 bytes defining the character in memory corresponds to one of the 8 lines of the character in the display, and each of the 8 bits in that byte is a dot in that line. If the bit is on (1), then the dot is white. If the bit is off (0), then the dot is black. For example, a circle with a dot in the middle could be defined as a character. It would require defining each of the 64 (8x8) dots as 64 (8x8) bits in memory. So

.....	00000000	binary	00	hex	0	decimal
••xxx•••	00111000		38		56	
•x•••x••	01000100		44		68	
x•••••x•	10000010		82		130	
x••x••x•	10010010		92		146	
x•••••x•	10000010		82		130	
•x•••x••	01000100		44		68	
••xxx•••	00111000		38		56	

The first 128 characters (00-7F, ASCII) are not under user control. The information required to display these characters is located in PROM at F800-FBFF (1K). The next 64 characters (80-BF, Exidy Graphics) can be programmed if desired, but they are already programmed to be standard keyboard graphics. The 64x8 (512) bytes for these are located at FC00-FDFF. This RAM can be changed at any time by the programmer to redefine these characters. However, the Monitor refreshes this area from its ROM every time a RESET

occurs, or whenever the video screen is cleared (e.g., when CLEAR is pressed, or when a Form Feed ASCII control is displayed). This will clobber any such modifications.

The last 64 characters (CO-FF) are completely under programmer control. They are always displayed as nonsense until they are "defined" by turning on and off the bits of the 8 bytes associated with the character. These bytes are in RAM from FE00 to FFFF (-512 to -1). For example, the character C0 (192) is a FE00-FE07 (-512 to -505), C1 (193) at FE08-FE0F (-504 to -497), C2 at FE10-FE17, and so on, until FF (255) is at FFF8-FFFF (-8 to -1). The formula to calculate where the 8 bytes in RAM begin for any of these 128 characters which can be programmed (80-FF) is (assume "c" is the character code of the character to be programmed):

$FC00 + (8 * (c - 80))$  hex, or  
 $(8 * (c - 128)) - 1024$  BASIC decimal

where "c" ranges from 80-FF (128-255).

For example, to print a "blot" (all dots on, a white square) on the screen followed by the above circle with the dot in the middle, the following BASIC program can be written. The blot will be made from the first programmable graphic 192, and the circle/dot will be 193:

```
10 FOR I=0 TO 7: REM 8 BYTES AT FE00 (-512) FOR BLOT
20 POKE -512+I,255: NEXT: REM TURN ON ALL BITS/DOTS
30 FOR I=0 TO 7: REM 8 BYTES AT FE08 (-504) FOR CHR #193
40 READ J: REM GET A BYTE VALUE FROM THE TABLE AS ABOVE
50 POKE -504+I,J: NEXT: REM TURN ON CORRECT DOTS
60 PRINT CHR$(192);CHR$(193): REM PRINT THE 2 NEW CHRS
70 DATA 0,56,68,130,146,130,68,56: REM DATA CHR #193
80 END
```

## EXIDY KEYBOARD ARCHITECTURE

The keyboard on the Exidy has a clever physical (hardware) and logical (software) architecture.

It actually resides on small parts of input and output ports FE (254). It is composed of a potential 80 keys, organized as sixteen rows of five columns each. For each one of the sixteen rows of possible keys (0-F, 0-15, output port FE bits 0, 1, 2, and 3) any one of the five columns of possible keys can be depressed (0-4, input FE bits 0, 1, 2, 3; and 4).

For example, row 0 column 0 is ESC, row 9 column 3 is a P, and row 15 column 4 is the = key on the numeric pad. Not all 80 possibilities are in use (about three are meaningless). Each of the valid possibilities can assume any one of five states:

1. When SHIFT is depressed — upper case, punctuation; no numerics or graphics; cursor, arrow keys operative.
2. When LOCK is depressed — this is a CAPS LOCK, so upper case letters, numerics, and punctuation are valid, but no graphics or cursor movement keys.

3. When CONTROL is pressed — this produces ASCII control characters, some numerics, and cursor movement; no graphics.
4. When GRAPHICS is pressed — this is standard Exidy keyboard graphics (codes 80-BF). If SHIFT is also pressed simultaneously, the programmable graphics codes CO-FF are used.
5. If none of the above are pressed — standard lower case and numerics and punctuation are used; no graphics or cursor movement.

The Monitor ROM area EC1E-EDFD contains the tables necessary to allow the keyboard input routine to translate the row/column of the key pressed into a 1-byte character codes, depending on which of the five states the keyboard is in. These tables are actually broken down into six tables total: the first is a what-to-do table to calculate the state etc., and the last five are the character codes for the five states.

## Performing Keyboard Input

To get keyboard input from the user from BASIC or Z80 Assembly Language without INPUT statements, a very useful subroutine can be used. In fact, this can be done such that the program sees each character as it is typed without having to wait (or ever get) a carriage return (RETURN). For example, a program can react and respond immediately to input commands as they are typed.

From BASIC, characters can be input with the following example assembly routines. Place this simple and relocatable Monitor keyboard routine driver interface at, say, location F0 (240). It can go anywhere, but F0 is a good start.

```
F0: CD15E0  SCAN: CALL QCKCHK ;Control-C pressed?
F3: C2FADF      JPNZ BASIC ;Yes, back to BASIC (warm)
F6: CD09E0      CALL RECEIVE ;No, get input character
F9: 28F5        JRZ SCAN ;Nothing yet, continue
FB: 32FF00      LD (CHR),A ;Got it, save at loc FF
FE: C9          RET ;Return after USR call
FF: 00          CHR: NOP ;Where byte stored for BASIC
```

The routine first checks to see if CTL-C, ESC, or RUN/STOP have been entered, meaning the user wants to quit. If so (Not Zero) back to READY level. If not, the current RECEIVE device (usually keyboard) is scanned for a character. If none (Zero), scanning continues. If found, the character is put at location FF (255). Control is then return to BASIC after the USR call. The following example BASIC program can use this routine:

```
10 PRINT "ENTER CHARACTER"
20 POKE 260,240: POKE 261,0: REM LOC 00F0 IS 240,0
30 Z=USR(Z): REM CALL SCAN
40 REM IF WE GET HERE LOC FF HAS A CHARACTER
50 A$=CHR$(PEEK(255))
60 IF A$="S" THEN STOP: REM STOP IF S ENTERED
70 PRINT A$: REM ECHO THE CHARACTER
80 GOTO 20: REM LOOP TILL S ENTERED
```

These are both simple routines that can be modified to be as fancy as possible.

From Z80 machine language there is no need to necessarily store the character in RAM. It is returned in the accumulator by the RECEIVE routine.

The above programs accept their input from the current RECEIVE device. To set this device the SET I=x command is used.

## Cursor Positioning

Cursor positioning is the process of moving the cursor (that underscore character) on the screen to locations other than where it usually is when standard BASIC or Monitor video output is done (e.g., PRINT, DUMP, etc.). This is very useful especially when data is to be placed on the screen but not in a line by line fashion. For example, if a graphic diagram is displayed and certain segments are to be labelled, the cursor can be moved directly to each one and the output generated in a random fashion on the screen. Also many times the usual output statements will destructively erase what is already on the screen. For example, if something is to be printed in the middle of a line but there is information already in the beginning of that line, an output statement will erase it. Cursor positioning to the middle will not.

To perform cursor positioning from Assembly Language or BASIC is quite simple:

1. Decide what line the cursor is to be on. There are 30 numbered 0-29. Call this "1". "L"
2. Decide what column of that line the cursor is to be on. There are 64 numbered 0-63 on each line. Call this "c".
3. Calculate  $64 \times 1$ . This is the offset from the beginning of the screen to the first column (0) of line 1. This is easy in BASIC ( $Q = 64 \times L$ ). In machine language, just shift 1 left six times, or, assuming 1 were in register E: \*

```
LD D,0 ;DE=01
LD B,6 ;TIMES TO SHIFT
X: SLA E ;SHIFT E
RL D ;SHIFT D
DJNZ X ;6 TIMES, DE=64x1
```

Or if 1 were in register pair HL, just execute the ADD HL,HL instruction six times in a row to double 1 six times, or multiply by 64.

4. Find the MWA. This is described in detail earlier. For the examples below, assume register IY points to the MWA for Assembly, and AD for BASIC.
5. At offset 68 hex (IY + 68 or AD + 104) is 2 bytes where  $64 \times 1$  is to be stored:

```
LD (IY+68),E
LD (IY+69),D
```

or in BASIC, POKE the low part (low byte) of the number  $64 \times 1$  ( $64 \times 1 \text{ MOD } 256$ ) into AD + 104, and POKE the high part (byte) of  $64 \times 1$  ( $\text{INT}(64 \times 1 / 256)$ ) at AD + 105. Now,  $64 \times 1 \text{ MOD } 256$  is just the remainder when  $64 \times 1$  is divided by 256, and this can be calculated as follows in BASIC:

```
905 L2=64*L
910 MD=L2 - INT(L2/256)*256
```

To do the POKEs, assuming AD is already pointing to the MWA:

```
915 POKE AD+104,MD
916 POKE AD+105,INT(L2/256)
```

6. At offset 6A in the MWA (IY + 6A, AD + 106) is 2 bytes where "c" is to be stored. If it were in register A:

```
LD (IY+6A),A
LD (IY+6B),O
```

or in BASIC

```
930 POKE AD+106,C
940 POKE AD+107,O
```

BASIC also requires you to put c at location 1BE (398) in the BCA:

```
950 POKE 398,C
```

7. Call the Monitor cursor move routine. This will replace the current cursor with the character which was at that spot ("undemeth" it), move the cursor to the requested spot and save the character there. From Z80:

```
CALL E9CC
```

From BASIC the USR technique must be used:

```
960 POKE 260,204: REM HEX CC
965 POKE 261,233: REM HEX E9
970 X=USR(X): REM CALL E9CC
```

8. Now a standard output statement like PRINT can be done and the output will begin at this new cursor location.

With this new technique, horizontal and vertical tabbing can also be done.

Horizontal tabbing may also be done in Basic directly with the use of the TAB(n) function.

Vertical tabbing may be done with Control-Z (down arrow) characters. For example, to tab to line 15 (0-29), home the cursor with a Control-Q — hex 11 — 17 decimal — and Control-Z fifteen times (Control-Z is hex 1A, decimal 16):

```
2220 PRINT CHR$(17); : REM HOME
2240 FOR I=1 TO 15
2260 PRINT CHR$(26); : REM DOWN ONE LINE
2280 NEXT
```

PRINT TAB(n) can then be used to tab horizontally on that line.

**EXIDY  
STANDARD  
BASIC**

## BASIC Floating Point Format

Numbers in BASIC are not integers. Fractions are allowed. Thus, the decimal point can move. For example, the decimal point "floats" when 13.25 is divided by 10 — 1.325. It is from this idea that the term "floating point" was derived.

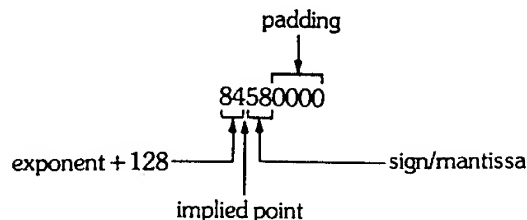
These numbers are stored by BASIC in four bytes of memory. Each number has three parts:

1. the sign (+ or -);
2. the "mantissa" (the actual number, but with the point shifted to the left of the leftmost 1 bit of the number). So the number 127 decimal (7F, 01111111) is a mantissa if it is thought of as .1111111;
3. the "exponent," which is how much the point had to be shifted in the number to produce the mantissa with the point at the left.

This all sounds very complex, but it actually is not. Let's take an example, say 13.5 decimal. In hex this would be equal to D.8 ( $13 + 8 \cdot 1/16$ ). Remembering that hex is just groups of four bits, the binary equivalent of 13.5 would be 1101.1000. To create a mantissa from this, we must shift the point (in this case, the "binary point," not the decimal point) to the left four places, producing .11011000. The exponent can now be calculated. It is always **positive** if the mantissa shift was to the **left**, **negative** if to the **right**, and **zero** if **no shift** was necessary. Thus, the exponent in this example would be +4 (four to the left). However, we are not quite done. Rather than worrying about how to express a negative number exponent, 128 decimal (hex 80) is always added to the exponent to produce the final result. Thus, the final exponent is 84 (132). Now we come to the sign. Since the digit to the far left of the mantissa is always 1 (because we shifted until that was the case), then the sign can be stored in this bit without losing any information. If the number is positive or zero, then the sign bit will be 0. If negative, then the sign bit will be a 1. So the mantissa for 13.5 .11011000 changes to .01011000. To assemble this number, first we put the exponent 84 then the mantissa filled out to the right to fill out the four bytes:

10000100 .01011000 00000000 00000000

Now if we ignore the point, since it is always in the same place, and convert to hex, we have:



If the original number were -13.5 instead, then nothing would change except the sign. That is the mantissa would change from .01011000 to .11011000, so the new number would be

84D80000

In the reverse direction, to convert floating point back to decimal, let's use 88FF4000 as an example:

1. Examine the exponent (88) and subtract hex 80 (128). In this example  $88 - 80 = 08$ . But this may produce a negative number.
2. Examine the mantissa with the implied point (.FF4000).
3. If the left bit (high order, the one next to the point) is on (it is), then the number is negative. Otherwise it is positive.
4. In either case, turn that bit on.
5. Shift the point according to the exponent from step 1 (08 here). If plus, shift right, if minus, left, if zero, no shift. Since we have +8, shift the point right 8 **bits**.

.111111110100000000000000

6. The number is now FF.4000, and with the sign, -FF.4000, or -255.25 decimal.

The only special case is the number 0. Here the exponent is 00. Other examples are:

1815	=	hex 717	=	8B62E000
1		1	=	81000000
-1		-1	=	81800000
-5		-8	=	80800000
0		0	=	0061000

The last idea that must be mentioned is that the number is actually stored in memory in **reverse**, so the number eemmnpp is stored pppnmee. For example, decimal 1815 in the above example:

00E628B

# BASIC CONTROL AREA

This is a discussion of the workarea in RAM used by BASIC, called the BASIC Control Area, or BCA. The BCA begins at address 100 (256), and has an overall appearance like

Table 5. BASIC Control Area	
100	BASIC Control Information
1D5	BASIC Program Source
a	BASIC Program Variables <i>PVF</i>
b	BASIC Program Arrays <i>IFA</i>
c	FREE SPACE
d	STACK
e	BASIC String Space <i>SSP</i>
	Monitor Stack
	MWA

In detail, RAM locations 100-14E (256-334) are copied from the BASIC ROM (address C258) when a BASIC Cold Start occurs (i.e., after Reset or a PP X command is entered). The BCA described below includes only those areas which are of direct use to the programmer. It is intentionally sketchy, especially due to the great number of fields.

*ROUTINES*  
 C7D0 - compute USR argument for 16 bit in EE  
 CF1A - complementary routine

## Address Description

100/256	Three-byte JUMP instruction to C06B (Warm Start). Done when PP command is entered without operands.
103/259	Three-byte JUMP to C7E5 default (displays "FC ERROR" message). This is the USR function hook. See BASIC Assembly interface section later for details.
145/325	Two-byte address of top of string space (letter "e" above) or the beginning of the BASIC stack. This is set by the BASIC CLEAR n command. <i>SSP</i>
147/327	BASIC line input buffer and Direct Mode execution line. <i>70</i>
18E/398	Current line column number.
1B1/433	Two-byte address of instruction in the BASIC program about to be executed when Control-C break is entered. This could be in the middle of a line of multiple statements separated by colons.
1B3/435	Two-byte BASIC line number of current line.
1B5/437	Two-byte address of the next <i>full</i> line to execute from the link pointer of the current line (see below).
1B7/439	Two-byte address of the end of the program and the beginning of the BASIC Program Variable Area (letter "a" above). <i>PVA</i>
1B9/441	Two-byte address of the end of the Variable Area and the start of the BASIC Program Array Area (letter "b" above). Whenever changes are made to the BASIC program (adding, deleting, updating lines) the above two addresses are used to define a new Variable and Array area below the new BASIC program. Thus, a program cannot be continued with old variable/array values once a change has been made. <i>PFA</i>
1BB/443	Two-byte address of the end of the Array Area and the pointer to free space (room for expansion — letter "C"). <i>FS</i>
1BD/445	Two-byte address of the last used data operand of a DATA statement so that the next READ will find the appropriate item. This is reset by a RESTORE command.
1BF/447	Four-byte input parameter (usually floating point format) to the USR function, and output parameter from the USR function. If USR (3.5) is called, 3.5 is passed to the subroutine in floating point. See a later section for BASIC/Assembly interfacing details.
1D5/469	Beginning of all BASIC programs.

## Format of BASIC String Variables and Arrays

A BASIC string variable is similar to a floating point variable. It is also six bytes long. It looks like:

Offset	Description
+0	Two-byte variable name. The high order bit is always 1.
+2	One-byte current length of the variable length string value.
+3	00
+4	Two-byte address of the string itself. It resides either in the string space or in the program statement itself (e.g., 1005 A\$="HI").

A string array is identical to a numeric array except for two very important features:

1. The high order bit of the array name is always 1.
2. The four byte value is not floating point format but the length/00/stringaddress fields described above. All dimensioning remains the same.

## Format of BASIC Program Statements

The first line of every BASIC program begins at location 1D5. All BASIC lines have the following variable length format:

Offset	Description
+0	Two-byte link pointer address of the next sequential full line in the program. This is independent of multiple statements on one line (separated by colons). The last line of the program points to location 0000 to indicate the end.
+2	Two-byte BASIC line number of the line in integer binary (a number between 0000 and FFF9, 0-65529).
+4	The BASIC statement(s), variable in length. Let us say they are "n" bytes long. Each BASIC "reserved word" such as GOTO, IF, END, DIM, PRINT, etc. is encoded here to a one-byte character not belonging to the ASCII character set (i.e., hex codes greater than 7F). This speeds up processing and saves program memory space. When the program is LISTed, these special bytes are decoded back into their corresponding reserved words.
+4+n	Byte of 00 indication the end of this line and beginning of the next.

## Format of BASIC Floating Point Variables and Arrays

A BASIC floating point variable resides in the BASIC Program Variable Area. Each one takes a constant six bytes:

Offset	Description
+0	Two-byte ASCII variable name. The high order bit is always 0. The letters are also reversed as usual.
+2	Four-byte floating point value currently held by this variable. See the format description earlier.

BASIC arrays all reside together after the variables in the BASIC Program Array Area. A floating point array is variable in length. It takes a minimum of seven bytes and looks like this: (Note: an array in Exidy BASIC can have any number of dimensions; call that number "n". Each can have any number of elements).

Offset	Description
+0	Two-byte array name. The high order bit is always 0. The letters are reversed.
+2	Two-byte total array length minus four (i.e., the length of the array starting after these two bytes). This is used to find the next array in the area quickly.
+4	One-byte number of dimensions (we called it n).
+5	Two-byte size (number of elements) in the first dimension.
+7	Two-bytes size of the second dimension (if any).
.	.
.	.
.	.
+5+2(n-1)	Two-byte size of the nth dimension.
+5+2n	Beginning of a list of contiguous four-byte floating point array elements. These are in Row order.

# BASIC to Z80 Assembly Language Interface

To call Z80 Assembly Language subroutines from Exidy BASIC, certain general conventions and procedures must be followed:

1. The machine language program must reside either in the first 256 bytes of memory (00-FF, 0-255 — usually a bad idea) or in the BASIC free space area described earlier. Either BASIC control, program, variables, arrays or strings, or Monitor/video control resides in the rest of memory. This is the only way a BASIC and machine language hybrid can coexist without complicated machinations such as putting the machine language routine right after the BASIC program and fooling BASIC into thinking that it is part of the program. The BASIC free space is the best and easiest choice. However there are some potential problems:

- a. Free space is dynamic. As the program changes, as variables/arrays are added or change size, the start of the free space moves. A machine language program placed too close to the end of the Array Area can get walked on. The end of the free space changes too, since the BASIC stack (or string space) will grow and shrink, especially with the CLEAR command. Since this change is usually not as radical as that of the start of the free space, I recommend putting the program close to the end of the free space. But there are now other considerations.
- b. The free space ends near HIMEM of the machine (where the BASIC stack is). This changes with each different Exidy size. So a generalized subroutine designed to run on any machine (probably with several BASIC programs) would either have to be relocatable (able to be moved without affecting anything), or there would have to be different versions of the program to run on different size machines. This of course would allow the BASIC program to use the maximum amount of free space. A subroutine designed for a *particular* BASIC program could be placed at the top of the free space as long as the BASIC program does not grow too much.
- c. If the program is placed at the end of the free space an excessive CLEAR n BASIC statement could kill it.
- d. Thus, no matter where the program is placed, certain restrictions have to be made to coexist with BASIC.

2. Assume a good location is found, and the Z80 program is written and relocated to that address in RAM. Assume this address to be 312A hex (12586). To call this subroutine from BASIC, it must already be in memory, and the USR function must be used. When BASIC executes it, it converts the argument to floating point and places this number in the four-byte USR parameter area at 1BF-1C2 (447-450). It then calls the subroutine at location 103 (259). For example, when the statement

2030 X=USR(25.7)

is executed, 25.7 is placed at 1BF and a CALL is made to 103.

3. Now, by default 103 contains the following Z80 instruction

JP C7E5

or in machine language — hex C3E5C7. This unconditional JUMP to the instruction at address C7E5 in BASIC ROM. This default subroutine prints the error message "FC ERROR" (function call invalid) and stops the program. To call *your* subroutine, you must change the JUMP instruction address to the address of the beginning of your program. Again the instruction after a BASIC Cold Start looks like

## Address Contents Description

103/259	C3	JUMP Z80 operation code
104/260	E5	Low part of address
105/261	C7	High part of address

Leave the C3 JUMP, but change the address. If your program was at 312A as we said, you must make the jump to 312A, or

JP 312A

or in machine language — hex C32A31. It is a good idea to change the two address bytes every time the subroutine is to be called. Use the BASIC POKE statement for this (which requires *decimal* operands). Put 2A (42) at location 104 (260), and put 31 (49) at location 105 (261):

10000 POKE 260,42  
10010 POKE 261,49  
10020 XX=USR(Y)

When the USR function is executed in line 10020, your routine at 312A will be called. It could use the value in variable Y placed at 1BF as input. It could also put another value back as output. This value will be returned to the BASIC statement as the "result" of the USR function. In the above example, the value returned will be placed in variable XX. Note that the short BASIC routine shown above can easily be made into a GOSUB subroutine by adding the statement

10030 RETURN

Thus, to call your routine you need only say

GOSUB 10000

4. To terminate your subroutine, one of four things can be done:
  - a. Return directly to the Monitor and exit BASIC altogether, e.g., for catastrophic errors. For Monitor Warm Start jump to address E003. For Cold Start use E000. The user will be shown the Monitor prompt (">").
  - b. For lesser errors detected, give an FC ERROR message, stop the program, and return to BASIC READY level. This is simply done by jumping to C7E5.
  - c. If errors are detected and your routines have displayed the error message(s), you can stop the program and exit directly to BASIC READY level. For a BASIC Warm Start jump to DFFA, for a Cold Start DFFD.
  - d. Of course you can return normally to BASIC so it will continue the program where it left off after the USR statement. This is simply done by the RET instruction. Fill in the parameter at 1BF first, if necessary.

Note that all the Monitor subroutines are available to the Z80 subroutine, including turning the tape on, reading a file, and turning it off; or getting input from the keyboard. See the section on Monitor Subroutines later.

Debugging of the Z80 routine is a little more difficult than debugging BASIC programs. BASIC loses control of the situation and of what you are doing while your routine is running, and can't "keep an eye out" for potential errors as it can within a BASIC program. Great care, desk checking, and modular programming are a must.

An assembly language routine can also use as input and output actual BASIC variables and arrays. Using the pointers in the BCA described earlier, the program can find the variable/array lists and scan for the one(s) with the correct name(s). The using the floating point or string formats, the values can be examined or changed.



# **EXIDY POWER-ON MONITOR**

## Monitor Workarea

This is a detailed description of the area of memory shown above at locations 1F91, 3F91, or 7F91, depending on the size of the machine.

The Monitor Workarea, hereafter called MWA, is the area in RAM used by the Exidy Monitor program to save important information needed for its successful operation. This area is always located right next to the Monitor Stack, and is always placed at the very top of available RAM space. For an 8K machine, the top of RAM is at 1FFF (8191), for 16K 3FFF (16383), and for 32K 7FFF (32767). This number, Himem, is placed by the Monitor in the two bytes at address F000-F001 (-4096 to -4095) in the video driver RAM space. Remember as with most micros, the two bytes are *reversed* in storage. For example, for a 16K Exidy, F000-F001 contains FF3F, not 3FFF. The address of the MWA can be obtained from this HIMEM address so that you don't have to worry about what size machine your programming is running on. To do this, you must get the HIMEM value at F000-F001 and subtract 6E (110) or add FF92 (-110). For example, in Z80 Assembly Language:

```
LD    HL,(F000) ;GET HIMEM
LD    BC,FF92  ;GET -110
ADD   HL,BC    ;HL POINTS TO THE MWA
```

Or in BASIC:

```
100 AD=256*PEEK(-4095)+PEEK(-4096)
110 IF AD>32767 THEN AD=AD-65536
120 AD=AD-110
```

There is also a Monitor subroutine designed to do this calculation for you. It is at address E1A2 (-7774). When CALLED, it puts the MWA address in Z80 register IY. Example:

```
CALL E1A2 ;IY POINTS TO THE MWA
```

A detailed map of the contents of the MWA will now be given. This will be in the same fashion as the overall memory map listed above, except that the addresses will be shown in a different form. First the offset in hex from the beginning of the MWA will be given. This can be used in Z80 Assembly Language as a displacement away from an index register such as IY, which points to the MWA. For example, if the displacement is listed as +41 to a particular field, then that field can be addressed in Z80 by (IY+41) or by 41(IY). The second part of the address is given as an absolute address of the field in RAM. Since the whole MWA moves dependent on the size of the machine, the first two hex digits of these addresses can change. The last two digits are always the same. So only these last two digits are listed. The first two will either be 1F (8K), 3F (16K), or 7F (32K). Note: if the user coldstarts the Sorcerer (Resets) with a size other than the above sizes (such as 21239 bytes, not even a whole multiple of a K) then the above addressing scheme is not applicable and only the displacement from the index register scheme may be used.

## Exidy Monitor Memory Map

To get an overall picture of how the Exidy utilizes the 64K of (possible) memory, a "memory map" is given.

Memory is cut up into pieces and each piece is used for a different purpose. In the map below the address of the first byte of each piece is listed along with the use of that area. The address is given in both hex and a form of decimal that is usable directly in BASIC with the PEEK and POKE commands. Note that some of these decimal numbers are negative. If the address exceeds 32767 (hex 7FFF), then BASIC requires that the "twos-complement" form of the number be used, or the negative form. For numbers greater than 7FFF, 65536 is subtracted from the number.

Be aware also that this is an *overall* wide angle view of memory. Detailed maps of certain areas (such as the Monitor Workarea and the BASIC Control Area) are included.

**Table 6. Monitor Memory Map**

Address	Description
0000	0 256-byte Z80 Restart space (RAM)
0100	256 User RAM start, begin BASIC Control Area (RAM)
1F00	7936 8K Monitor Stack end (8K machines) (RAM)
3F00	16128 16K
7F00	32512 32K
1F90	8080 8K Monitor Stack start (8K machines) (RAM)
3F90	16272 16K
7F90	32656 32K
1F91	8081 8K Monitor Workarea start (8K machines) (RAM)
3F91	16273 16K
7F91	32657 32K
1FFF	8191 8K End User RAM (8K machines) (RAM)
3FFF	16383 16K
7FFF	32767 32K
C000	-16384 Begin 8K ROM PAC (e.g., begin BASIC) (ROM)
E000	-8192 Begin 4K Monitor Program (ROM)
F000	-4096 128-byte video driver space (RAM)
F080	-3968 1920-byte video screen (64x30) (RAM)
F800	-2048 1K standard Exidy ASCII alphanumerics (00-7F) (PROM)
FC00	-1024 512-byte Exidy keyboard standard graphics character set, accessed by depressing GRAPHICS key, character codes hex 80-BF (128-191) (RAM)
FE00	-512 512-byte User Programmable graphics character set, accessed by depressing SHIFT and GRAPHICS keys, codes hex CO-FF (192-255) (RAM)
FFFF	-1 End Exidy address space (64K)

**Table 7. Monitor Workarea**

Address	Description	Address	Description
+00 91 -110	60-byte Monitor command input buffer. Any command entered from the current RECEIVE device (SET I=x) such as the keyboard, serial or parallel ports is placed in this area. It is left-justified, and terminated by an ASCII carriage return character (hex code 0D, 13 decimal, hereafter called a CR). The Monitor subroutine at E13A (-7878) builds this buffer from the input.	+47 D8 71 -39	Beginning of the 16-byte tape output file header area. The first 5 bytes here contain the 5-character ASCII file name as entered on the SAVE or CSAVE command. It is left justified and padded to the right with ASCII blanks (code 20, 32 decimal).
+3C CD 63 -50	Port FE interface status.	+4C DD 76 -34	File header id, usually hex 55.
+3D CE 61 -49	Serial interface and dual cassette interface baud rate save area. 1200 baud is indicated by hex 40, 300 baud by the value 00. Serial port or cassette baud rates are set to the default of 1200 baud (hex 40) by the Monitor COLD Reset routine (at E000, -8192) and by the Monitor USER Reset entry point (at E003, -8189). Such a coldstart is done, for example, when the RESET keys are depressed. This byte is also set by the SET T=0 and SET T=1 commands (at Monitor routines at E5A2, -6750).	+4D DE 77 -33	File type. Usually C2 (194) for a BASIC save file. If the high order bit (80, 128 decimal) is on, the file cannot be automatically executed with the LOADG command. This is set by the SET F=xx command.
-3E CF 62 -48	SEND delay time. This value is used to delay before a SEND (to video, serial, or parallel) is done. The actual delay is about 1500 times this value machine cycles. This delay can therefore range from 0 to approximately 400000 cycles. The value is set by the SET S=n command.	+4E DF 78 -32	2-byte length of the file in bytes.
+3F D0 65 -47	Current SEND routine address. The default address set by COLD starts is the video routine at E9F0 (-5648). It can be changed by the SET O=x command.	+50 E1 80 -30	2-byte program loading address. For BASIC files, this is always 01D5 (469) because BASIC programs always start at that address. See the BASIC Control Area description following. For other programs such as those in machine language, this address is the "ssss" of the command "SAVE name ssss eeee."
+41 D2 65 -45	Current RECEIVE routine address. The default is set by COLD starts to be the keyboard routine at EB1C, -5348. It can be changed by the SET I=x command.	+52 E3 82 -28	2-byte program "go-address" for auto execution files. The Monitor will automatically begin execution of the program at this address with the LOADG command. This address is set by the SET X=nnnn command.
+43 D4 67 -43	Batch mode status. 00=normal input, nonzero=batch mode. This byte is used by the Monitor command input routine (E142) to determine whether commands are to be gotten from the RECEIVE device or from the batch tape serial port. The OVER command turns this off and the BATCH command turns this on.	+54 E5 84 -26	3 bytes of reserved space, ending the output tape header.
+44 D5 67 -42	Monitor output prompt character. The default is the character ">" or ASCII code 3E (62) set by COLD starts. It can be changed by the PROMPT x command. It is output to the SEND device every time a Monitor input command is being requested (at EOED, -7955).	+57 E8 87 -23	16-byte tape input header area. The format is identical to that of the area at +47. This area is filled in from reading the tape for commands such as CLOAD, LOAD, FILES, and so on.
+45 D6 69 -41	Tape status, baud rate, motor control save area. This is zeroed when the tape(s) is turned off, and otherwise remembers the status of the tape baud rates (00=300, 40=1200) and motor controls (10=motor #1 on, 20=motor #2 on).	+67 F8 103 -7	Character under the cursor. Since the cursor is an underscore character (ASCII code 5F, 95 decimal), it actually <b>replaces</b> the character at the cursor location. This hidden character is saved to be put back when the cursor is moved. The save is done by E9CC (-5684), and it is replaced by E9E8 (-5656).
+46 D7 70 -40	Tape input and output CRC (Cyclic Redundancy Check). The CRC is used to check whether the data has been transmitted successfully to/from the tape. This technique is described in detail in a subsequent section.	+68 F9 104 -6	2-byte line number where the cursor is times 64. This ranges from 0x64 (0) to 29x64 (1856), and is the offset from the beginning of the screen to the cursor line start.
		+6A FB 106 -4	2-byte cursor column number (0-63). When added to +68 the actual cursor offset into the screen is found.
		+6C FD 108 -2	Last character entered from the keyboard. This is used for the processing of the REPT (repeat) key logic. This character is entered to the keyboard input routine about every 30000 machine cycles as long as the REPT key is depressed. It is always the last key entered, and is saved and used by the keyboard processing routine at EB1C (-5348).
		+6D FE 109 -1	Two bytes of reserved space. This brings us to the end of the MWA, and in fact the end of user RAM.

# Monitor Subroutines

The Exidy ROM Monitor is just packed with very well-written and useful subroutines which can be called from BASIC and assembly language. All are resident in the 4K ROM between locations E000 and EFFF. This is a brief description of all the useful routines, and how to interface them. Here the address will be given in hex of course, but will also be given as a two-part decimal number in the order necessary to POKE into the USR JUMP vector at locations 260-261.

**Table 8. Monitor Subroutines**

Address	Description	Address	Description
E000 0,224	Monitor Cold Start (on RESET).	E1A2 162,225	Will find MWA and put the address in IY without causing screen flicker (only does so during vertical retrace on the TV to avoid DMA conflicts).
E003 3,224	Monitor Warm Start (on BYE command).	E1BA 186,225	SENDLINE: sends an entire line to the SEND device. HL points to the line, which must end in a 00. LFs are always sent when CRs are found.
E006 6,224	Monitor User Cold Start — similar to E000 except HL is input containing what the user wants to use as HIMEM.	E1C9 201,225	ERROR: sends "ERROR" followed by the diagnostic message (which is pointed to by HL).
E009 9,224	RECEIVE: returns NZ and a character from the current RECEIVE device in the accumulator (A), or Z if no character yet.	E1D4 212,225	OVER command processor (CP). Handles all work necessary for the OVER command.
E00C 12,224	SEND: sends character in A to the current SEND device.	E1E8 232,225	Sends 4-byte ASCII equivalent of the 2-byte integer in DE. If DE = 3F29, then "3F29" is sent.
E00F 15,224	SERIAL IN: reads a character into A from the serial input device or from cassette tape.	E1ED 237,225	Send 2-byte ASCII of byte in A.
E012 18,224	SERIAL OUT: writes character from A to serial/tape.	E205 5,226	Send a CR followed by a LF, CRLF.
E015 21,224	QCKCHK: returns NZ if Control-C or ESC (RUN/STOP) is depressed, otherwise it returns Z.	E23D 61,226	Convert a 1-4 byte ASCII hex number (pointed to by HL) into DE. If HL points to A93 followed by a "Monitor Delimiter" (e.g., blank, CR, etc.), then DE will contain 0A93. This is the reverse process of the routine at E1E8.
E018 24,224	KEYBOARD: the RECEIVE routine if SET I=K (default) See E009.	E2D2 210,226	Send as many blanks as the number in B.
E01B 27,224	VIDEO: the SEND routine if SET O=V (default). See E00C.	E4D3 211,228	DUMP CP
E01E 30,224	PARALLEL IN: the RECEIVE routine if SET I=P.	E538 56,229	ENTER CP
E021 33,224	PARALLEL OUT: the SEND routine if SET O=P.	E562 98,229	MOVE CP
E993 147,233	CENTRONICS OUT: the SEND routine for SET O=L.	E597 151,229	GO CP
E024 36,224	CASSETTE MOTOR CONTROL ON: will turn motor on and set the baud rate of the requested cassette. MWA+3D must contain the baud rate (00=300, 40=1200) and reg B must contain the cassette number (1 or 2).	E5A2 162,229	SET CP
E027 39,224	CASSETTE OFF: turns off both tapes.	E638 56,230	SAVE CP
E02A 42,224	TAPE SAVE: Save memory onto tape. MWA+50, MWA+51 must contain the memory address where SAVEing is to start. It must also be pushed on the stack. DE must contain the ending address. HL must point to a byte containing a CR (hex 0D). MWA+47 through MWA+4B must contain the ASCII file name; MWA+4D must contain the file type; MWA+52,MWA+53 the GO address, if any.	E6B9 185,230	FILES CP
E02D 45,224	TAPE LOAD: load a file into memory from tape. MWA+47 through MWA+4B must contain the file name to load. If a LOADG is to be done, a Z flag must be on the stack, otherwise an NZ flag. Then if the program name is specified, put NZ in the flags, otherwise Z (i.e., load the next file on the tape).	E78A 138,231	LOAD CP
E13A 58,225	MONITOR INPUT: will put the command in the command input buffer at MWA+0. IY must point to the MWA. MWA+43 must contain 0 (not Batch).	E845 69,232	PROMPT CP
		E858 88,232	BATCH CP
		E85C 92,232	CREATE CP
		E884 132,232	LIST CP
		E8A1 161,232	TEST CP
		E98A 138,233	PP CP
		E9B1 177,233	Clear the video screen and refresh/rewrite the graphics character set at FC00.
		E9CC 204,233	Move the cursor to line/column specified in the MWA. See cursor positioning described previously.
		E9D6 214,233	Find the cursor. HL is set to the screen address (which starts at F080) and DE is set to the column number.
		EB10 16,235	Refresh character set at FC00.
		EC1E 30,236	Keyboard input tables (to EDFD). See keyboard section.
		EDFE 254,237	Character set for the 64 standard graphics 80-BF to be copied to FC00.

# **MONITOR LISTINGS**

```
0000          0002 ;
0000          0003 ;
0000          0004 ;
0000          0005 ; *****
0000          0006 ; *
0000          0007 ; *   EXIDY STANDARD MONITOR   *
0000          0008 ; *
0000          0009 ; *****
0000          0010 ;
0000          0011 ;
0000          0012 ;
0000          0013 ;
0000          0014 ;
0000          0015 ;
0000          0016 ;   DEVELOPED FOR EXIDY INC.
0000          0017 ;
0000          0018 ;   BY   JOHN K. BORDERS JR.
0000          0019 ;
0000          0020 ;
0000          0021 ;
0000          0022 ; Z80 BASED MONITOR SOFTWARE
0000          0023 ; WITH FULL CASSETTE AND VIDEO
0000          0024 ; DRIVER ROUTINES. SELF-SEEKING
0000          0025 ; RAM STORAGE AND STACK ROUTINES.
0000          0026 ;
0000          0027 ;
0000          0028 ;
0000          0029 ;
0000          0030 ;
0000          0031 ; VERSION 1.0   DATED:  7/26/78
0000          0032 ;
0000          0033 ;
0000          0034 ;
0000          0035 ;
0000          0036 ;
```

```

0000      0038 ;
0000      0039 ;
0000      0040 ;      EQUATE TABLE
0000      0041 ;
0000      0042 ;
0000      0043 ;*****
0000      0044 ;
0000      0045 ; ASCII EQUATES
0000      0046 ;
      000D      0047 CR:      EQU      0DH      ;CARRIAGE RETURN
      000A      0048 LF:      EQU      0AH      ;LINE FEED
      001B      0049 ESC:      EQU      1BH      ;ESCAPE
      0001      0050 CNTRLA: EQU      'A'-40H
      0003      0051 CNTRLB: EQU      'B'-40H
      0008      0052 CNTRLH: EQU      'H'-40H
      0011      0053 CNTRLQ: EQU      'Q'-40H
      0013      0054 CNTRLS: EQU      'S'-40H
      0017      0055 CNTRLW: EQU      'W'-40H
      001A      0056 CNTRLZ: EQU      'Z'-40H
      007F      0057 RUBOUT: EQU      7FH      ;RUB OUT
      0020      0058 SPACE: EQU      20H      ;SPACE
0000      0059 ;
0000      0060 ; RAM POINTERS
0000      0061 ;
      F000      0062 RAMTOP: EQU      0F000H      ;POINTER STORE
      0000      0063 RAM:      EQU      0000H      ;START OF RAM
      0000      0064 BUFFER: EQU      0      ;INPUT BUFFER
      003C      0065 LINELN: EQU      BUFFER+60      ;LINE LENGTH
      003D      0066 TAPES:  EQU      LINELN+1      ;TAPE RATE
      003E      0067 SPEEDS: EQU      TAPES+1      ;DISPLAY SPEED
      003F      0068 OUTADD: EQU      SPEEDS+1      ;OUTPUT ADDRESS
      0041      0069 INADD:  EQU      OUTADD+2      ;INPUT ADDRESS
      0043      0070 BATCHF: EQU      INADD+2      ;BATCH FLAG
      0044      0071 PROMPT: EQU      BATCHF+1      ;PROMPT CHARACTER
      0045      0072 CMTRFG: EQU      PROMPT+1      ;CASSETTE MOTOR FLAG
      0046      0073 CRCBYT: EQU      CMTRFG+1      ;CRC BYTE
      0047      0074 CHEAD:  EQU      CRCBYT+1      ;COMMAND HEADER
      0057      0075 THEAD:  EQU      CHEAD+16      ;TAPE HEADER
0000      0076 ; VIDEO SCREEN EQUATES
      00F8      0077 TOPHRG: EQU      0F8H
      F000      0078 SCREEN: EQU      0F000H
      F080      0079 VID:      EQU      SCREEN+128
      FB00      0080 TOP:      EQU      SCREEN+2048
      0067      0081 VDHLN:  EQU      THEAD+16      ;CHAR HOLD
      0068      0082 LINE:    EQU      VDHLN+1      ;LINE #
      006A      0083 CHR:      EQU      LINE+2      ;CHAR #
      006C      0084 LSTKEY:  EQU      CHR+2      ;LAST KEY PRESSED
      0010      0085 HEADLN:  EQU      16      ;HEADER LENGTH
      006E      0086 STORE:   EQU      LSTKEY+2      ;END OF EQU TABLE
0000      0087 ;
0000      0088 ; CASSETTE HEADER EQUATES
0000      0089 ;
      0000      0090 HNAME:   EQU      0
      0006      0091 HTYPE:   EQU      6
      0007      0092 HSIZE:   EQU      7
      0009      0093 HADDR:   EQU      9
      000B      0094 HEXEQ:   EQU      11

```

```

0000          0096 ;
0000          0097 ;
0000          0098 ;
0000          0099 ;
0000          0100 ;
0000          0101          ORG      0E000H
E000          0102 ;
E000          0103 ;
E000          0104 ;
E000          0105 ;          JUMP TABLE INTO MONITOR
E000          0106 ;
E000          0107 ;
E000          0108 ;
E000  C3 62 E0 0109 COLD:  JP      INITC      ;COLD START
E003  C3 E8 E0 0110 WARM:  JP      INITW      ;WARM START
E006  C3 77 E0 0111 USER:  JP      INITU      ;USER START
E009  C3 30 E0 0112 RECEVE: JP      CHRIN      ;INPUT CHARACTER
E00C  C3 45 E0 0113 SEND:  JP      CHROUT     ;OUTPUT CHARACTER
E00F  C3 DA E2 0114 INTAPE: JP      TAPEIN     ;TAPE INPUT
E012  C3 EE E2 0115 OUTAPE: JP      TAPOUT     ;TAPE OUTPUT
E015  C3 D1 EA 0116 QUIKCK: JP      QUIK       ;CNTRL C CHECK
E018  C3 1C EB 0117 KEYBRD: JP      CHRIN1     ;KEYBOARD INPUT
E01B  C3 F0 E9 0118 VIDEO:  JP      CHROT1     ;VIDEO OUTPUT
E01E  C3 76 E7 0119 PARLIN: JP      PARIN      ;PARALLEL INPUT
E021  C3 7F E7 0120 PARLOT: JP      PAROUT     ;PARALLEL OUTPUT
E024  C3 8A E2 0121 CMOTON: JP      MOTRON     ;TURN CASSETTE MOTOR ON
E027  C3 AF E2 0122 CMOTOF: JP      MTROFF     ;TURN CASSETTE MOTOR OFF
E02A  C3 5A E6 0123 BASSAV: JP      SAVBAS     ;ENTRY FOR BASIC CSAVE
E02D  C3 99 E7 0124 BASLOD: JP      LODBAS     ;ENTRY FOR BASIC CLOAD
E030          0125 ;
E030  FD E5 0126 CHRIN:  PUSH  IY          ;WE DESTROY THESE
E032  E5 0127          PUSH  HL
E033  CD A2 E1 0128          CALL  GETIY
E036  21 41 E0 0129          LD    HL,CHRINR  ;FOR RETURN
E039  E5 0130          PUSH  HL
E03A  FD 6E 41 0131          LD    L,(IY+INADD);GET ADDRESS
E03D  FD 66 42 0132          LD    H,(IY+INADD+1)
E040  E9 0133          JP    (HL)          ;GO DO IT
E041  E1 0134 CHRINR:  POP   HL          ;RESTORE
E042  FD E1 0135          POP   IY
E044  C9 0136          RET
E045          0137 ;
E045  FD E5 0138 CHROUT: PUSH  IY          ;WE DESTROY THESE
E047  E5 0139          PUSH  HL
E048  F5 0140          PUSH  AF
E049  CD A2 E1 0141          CALL  GETIY
E04C  FD 66 3E 0142          LD    H,(IY+SPEEDS);GET DISPLAY SPEED
E04F  2E 01 0143          LD    L,1          ;FINISH OFF
E051  2B 0144 OUTDLY:  DEC   HL          ;DELAY
E052  7C 0145          LD    A,H          ;ARE WE THROUGH?
E053  B5 0146          OR    L
E054  20 FB 0147          JR    NZ,OUTDLY ;NOPE-
E056  F1 0148          POP   AF          ;GET 'EM BACK
E057  21 41 E0 0149          LD    HL,CHRINR
E05A  E5 0150          PUSH  HL
E05B  FD 6E 3F 0151          LD    L,(IY+OUTADD)
E05E  FD 66 40 0152          LD    H,(IY+OUTADD+1)
E061  E9 0153          JP    (HL)

```



```

E062      0154 ;
E062      0155 ;
E062      0156 ;
E062      0157 ;   INITIALIZE ROUTINES,
E062      0158 ;
E062      0159 ;
E062      0160 ;
E062      0161 ;
E062      0162 ;
E062      0163 ;
E062      0164 ;   INITC = COLD START - FINDS
E062      0165 ;   TOP OF RAM AND SETS
E062      0166 ;   STACK AND STORAGE THERE
E062      0167 ;   INITW = WARM START - USES
E062      0168 ;   STACK FROM INITC
E062      0169 ;   INITU = USER START - USES
E062      0170 ;   HL FROM USER AS TOP OF
E062      0171 ;   RAM LIKE INITC
E062      0172 ;
E062      0173 ;
E062      0174 ;
E062      0175 INITC: EQU $
E062 3E FF      0176      LD      A,OFFH      ;INITIALIZE CASSETTE
E064 D3 FD      0177      ?OUT    OFDH,A
E066 16 00      0178      LD      D,0
E068 21 00 00    0179      LD      HL,RAM      ;POINT BEG RAM
E06B 7E         0180 INITC2: LD      A,(HL)      ;GET IT
E06C 46         0181      LD      B,(HL)      ;TWICE
E06D 2F         0182      CPL              ;TURN AROUND
E06E 77         0183      LD      (HL),A      ;PUT BACK
E06F BE         0184      CP      (HL)      ;& CHK IT
E070 70         0185      LD      (HL),B      ;PUT REAL BACK
E071 23         0186      INC     HL          ;POINT NEXT
E072 28 F7      0187      JR      Z,INITC2    ;LOOP IF GOOD
E074 2B         0188      DEC     HL          ;ADJUST
E075 2B         0189      DEC     HL          ; H & L
E076 01         0190      DB      1          ;LXI B
E077          0191 ;
E077          0192 ;   USER START ENTRY POINT
E077          0193 ;
E077          0194 INITU: EQU $
E077 16 01      0195      LD      D,1
E079          0196 INITU1: EQU $
E079 22 00 F0    0197      LD      (RAMTOP),HL ;USER IS HERE
E07C FD 2A 00 F0 0198      LD      IY,(RAMTOP)
E080 01 92 FF    0199      LD      BC,0-STORE
E083 FD 09      0200      ADD     IY,BC      ;DO IT
E085 FD F9      0201      LD      SP,IY      ;GET A STACK
E087 CD D1 EA    0202      CALL    QUIK      ;SEE IF WARM RESET
E08A C2 FA DF    0203      JP      NZ,PWARM   ;YES-GO DO IT
E08D 7D         0204      LD      A,L        ;CLEAR RAM!
E08E FD E5      0205      PUSH    IY
E090 E1         0206      POP     HL          ;GET BEGINNING
E091 36 00      0207 INITU2: LD      (HL),0    ;MAKE ZERO
E093 23         0208      INC     HL          ;NEXT
E094 BD         0209      CP      L          ;THRU?
E095 20 FA      0210      JR      NZ,INITU2   ;NO-KEEP GOIN'
E097 FD 36 44 3E 0211      LD      (IY+PROMPT),'>' ;INIT PROMPT

```

```

E09B FD 36 3D 40      0212      LD      (IY+TAPES),40H;SET FOR 1200 BAUD
E09F D5                0213      PUSH   DE          ;WE DESTROY
EOA0 CD B1 E9          0214      CALL   VIDINT      ;INIT VIDEO BOARD
EOA3 D1                0215      POP    DE          ;GET BACK
EOA4 21 1C EB          0216      LD      HL,CHRIN1 ;POINT KEYBOARD
EOA7 FD 75 41          0217      LD      (IY+INADD),L
EOAA FD 74 42          0218      LD      (IY+INADD+1),H;PUT AWAY
EOAD 21 F0 E9          0219      LD      HL,CHROT1 ;POINT VIDEO
EOB0 FD 75 3F          0220      LD      (IY+OUTADD),L
EOB3 FD 74 40          0221      LD      (IY+OUTADD+1),H
EOB6 3A FD DF          0222      LD      A,(PCOLD) ;SEE IF PROM PACK IS IN
EOB9 FE C3             0223      CP      0C3H        ;IS THE "JUMP" THERE?
EOBB 20 0B             0224      JR      NZ,INITU3 ;NO-
EOBD 3A FA DF          0225      LD      A,(PWARM) ;HOW ABOUT THIS ONE?
EOC0 FE C3             0226      CP      0C3H
EOC2 20 04             0227      JR      NZ,INITU3
EOC4 15                0228      DEC     D
EOC5 C3 8D E9          0229      JF      PROMP1
EOC8 21 62 E3          0230 INITU3: LD      HL,HEDING ;POINT MSG
EOCB CD BA E1          0231      CALL   MSGOUT
EOCE ED 5B 00 F0       0232      LD      DE,(RAMTOP)
EOD2 CD E8 E1          0233      CALL   ADDOUT      ;PRINT RAM TOP
EOD5 21 BC E3          0234      LD      HL,HEAD2
EOD8 CD BA E1          0235      CALL   MSGOUT      ;FINISH
EODB FD E5             0236      PUSH   IY          ;PUT STACK
EODD D1                0237      POP    DE          ; IN DE
EODE 1B                0238      DEC     DE          ;ADJUST
EODF CD E8 E1          0239      CALL   ADDOUT      ;PRINT IT
EOE2 21 D5 E3          0240      LD      HL,HEAD3 ;LAST ONE
EOE5 CD BA E1          0241      CALL   MSGOUT
EOE8                  0242 ;
EOE8                  0243 ;      WARM START ENTRY POINT
EOE8                  0244 ;
EOE8 E0E8              0245 INITW: EQU     $
EOE8 CD A2 E1          0246      CALL   GETIY      ;GO GET A VALID IY FROM R4

```

```

EOEB      0248 ;
EOEB      0249 ;
EOEB      0250 ;
EOEB      0251 ;   BEGINNING OF MAIN PROGRAM
EOEB      0252 ;
EOEB      0253 ;
EOEB      0254 ;
EOEB      0255 ;
EOEB      0256 ;
EOEB      0257 START: EQU    $           ;START MAINLINE
EOEB      FD F9      0258      LD      SP,IY       ;REDO STACK
EOED      CD 05 E2    0259      CALL   CRLF        ;FRESH LINE
EOFO      FD 7E 44    0260      LD      A,(IY+PROMPT);LOAD PROMPT
EOF3      CD 45 E0    0261      CALL   CHROUT      ;& OUT
EOF6      CD 3A E1    0262      CALL   LINEIN      ;GET A LINE
EOF9      FD E5      0263      PUSH   IY          ;MOVE IY
EOFB      E1         0264      POP    HL          ;   TO HL
EOFC      CD 25 E2    0265      CALL   SCAN        ;SKIP DELIMS
EOFF      CA 34 E1    0266      JP     Z,ERRCMD    ;UH-OH NONE!
E102      DD 21 12 E3 0267      LD      IX,TABLE    ;POINT CMD TBL
E106      E5         0268 MAIN1:  PUSH   HL        ;SAVE'EM
E107      DD E5      0269      PUSH   IX
E109      06 02      0270      LD      B,2         ;CHECK 2 CHRS
E10B      DD 7E 00    0271 MAIN2:  LD      A,(IX)     ;LOAD FRM TABLE
E10E      BE         0272      CP      (HL)        ;COMPARE?
E10F      20 12      0273      JR     NZ,MAIN4     ;NO-TRY NEXT
E111      23         0274      INC     HL          ;CHECK NEXT CHR
E112      DD 23      0275      INC     IX
E114      10 F5      0276      DJNZ   MAIN2        ;LOOP FOR 2
E116      D1         0277      POP     DE          ;CLEAR STACK
E117      D1         0278      POP     DE          ;I/O PNTR
E118      01 EB E0    0279      LD      BC,START    ;PUSH RETURN
E11B      C5         0280      PUSH   BC
E11C      DD 6E 00    0281 MAIN3:  LD      L,(IX)     ;LOAD
E11F      DD 66 01    0282      LD      H,(IX+1)    ; JUMP
E122      E9         0283      JP      (HL)        ; ADDRESS
E123      DD E1      0284 MAIN4:  POP     IX        ;RESET
E125      E1         0285      POP     HL          ; POINTERS
E126      DD 23      0286      INC     IX          ;NEXT ENTRY
E128      DD 23      0287      INC     IX          ; IN
E12A      DD 23      0288      INC     IX          ; TABLE
E12C      DD 23      0289      INC     IX
E12E      DD 7E 00    0290      LD      A,(IX)     ;IS IT
E131      B7         0291      OR      A          ; THE END?
E132      20 D2      0292      JR     NZ,MAIN1     ;NO-CONTINUE
E134      21 E6 E3    0293 ERRCMD: LD      HL,IVCMMSG
E137      C3 C9 E1    0294      JP      WHAT

```

```

E13A      0296 ;
E13A      0297 ;
E13A      0298 ;
E13A      0299 ;   LINE INPUT ROUTINE
E13A      0300 ;
E13A      0301 ;
E13A      0302 ;
E13A      0303 ;
E13A      0304 ;   THE FOLLOWING ARE COMMAND CHRS:
E13A      0305 ;           <CR> = END LINE
E13A      0306 ;           RUB  = BACKSPACE
E13A      0307 ;           @   = START OVER
E13A      0308 ;   ALL OTHER CHARS SIMPLY INPUT
E13A      0309 ;
E13A      0310 ;
E13A      0311 ;
E13A      E13A 0312 LINEIN: EQU    $
E13A      FD E5 0313          PUSH  IY          ;MOVE IY
E13C      E1    0314          POP    HL          ;   TO HL
E13D      3E 3C 0315          LD     A,LINELN    ;SET LINE
E13F      85    0316          ADD    L          ;   LENGTH AND
E140      4F    0317          LD     C,A         ;   START IN
E141      45    0318          LD     B,L         ;   IN BC
E142      FD 7E 43 0319          LD     A,(IY+BATCHF);GET BATCH FLAG
E145      B7    0320          OR     A          ;TEST IT
E146      20 39 0321          JR     NZ,LINE3    ;GO BATCH IT
E148      CD 30 E0 0322 LINE1: CALL  CHRIN      ;GET
E14B      28 FB 0323          JR     Z,LINE1
E14D      CB 7F 0324          BIT     7,A
E14F      20 1D 0325          JR     NZ,LINE2A
E151      FE 0D 0326          CP     CR
E153      28 12 0327          JR     Z,LINE2
E155      FE 20 0328          CP     SPACE      ;CONTROL CHR?
E157      38 15 0329          JR     C,LINE2A   ;YES!
E159      FE 7F 0330          CP     RUBOUT     ;RUB?
E15B      28 18 0331          JR     Z,BKSPC    ;YES!
E15D      FE 40 0332          CP     '@'       ;NEW LINE?
E15F      20 06 0333          JR     NZ,LINE2   ;NO-GO ON
E161      CD 05 E2 0334          CALL  CRLF     ;NEXT LINE
E164      C3 3A E1 0335          JP     LINEIN   ;OVER
E167      77    0336 LINE2:  LD     (HL),A     ;PUT AWAY
E168      23    0337          INC     HL        ;POINT NEXT
E169      FE 0D 0338          CP     CR        ;CAR RET?
E16B      CA 05 E2 0339          JP     Z,CRLF   ;YES-DO IT & RET
E16E      CD 45 E0 0340 LINE2A: CALL  CHROUT    ;PRINT IT
E171      79    0341          LD     A,C       ;TOO MANY
E172      BD    0342          CP     L         ;   CHARS?
E173      20 D3 0343          JR     NZ,LINE1   ;NO-CONTINUE
E175      0344 ;
E175      0345 ;   BACKSPACE ROUTINE
E175      0346 ;
E175      78    0347 BKSPC:  LD     A,B         ;ARE YOU
E176      BD    0348          CP     L         ;   AT BEG?
E177      28 CF 0349          JR     Z,LINE1    ;YES-IGNORE
E179      3E 08 0350          LD     A,CNTRLH
E17B      CD 45 E0 0351          CALL  CHROUT
E17E      2B    0352          DEC     HL        ;DELETE CHR
E17F      18 C7 0353          JR     LINE1     ;CONTINUE

```

```

E181      0354 ;
E181      0355 ;
E181      0356 ;
E181 06 01 0357 LINE3: LD B,1
E183 E5 0358 PUSH HL ;SAVE BEGINNING
E184 CD 8A E2 0359 CALL MOTRON
E187 CD 59 E7 0360 CALL TAPWT ;WAIT FOR NULLS
E18A CD DA E2 0361 LINE4: CALL TAPEIN ;GET
E18D CA D4 E1 0362 JP Z,FINISH
E190 77 0363 LD (HL),A ;PUT
E191 23 0364 INC HL ;NEXT
E192 FE 0D 0365 CP CR ;IS IT?
E194 20 F4 0366 JR NZ,LINE4
E196 CD 4E E7 0367 CALL CKCRC
E199 CD AF E2 0368 CALL MTROFF ;TURN OFF
E19C 36 00 0369 LD (HL),0
E19E E1 0370 POP HL
E19F C3 BA E1 0371 JP MSGOUT
E1A2 0372 ;
E1A2 0373 ;
E1A2 0374 ;
E1A2 0375 ; CREATES A IY FROM RAMTOP IN F000
E1A2 0376 ;
E1A2 0377 ;
E1A2 E1A2 0378 GETIY: EQU $
E1A2 C5 0379 PUSH BC ;I NEED
E1A3 F5 0380 PUSH AF
E1A4 DB FE 0381 SEEIFR: IN A,0FEH ;SEE IF SCREEN IS READY
E1A6 CB 6F 0382 BIT 5,A
E1A8 28 FA 0383 JR Z,SEEIFR
E1AA F1 0384 POP AF
E1AB 06 08 0385 LD B,WINPK5 ;LOAD WAIT CONSTANT
E1AD 10 FE 0386 WFBTZ: DJNZ WFBTZ ;WAIT FOR B TO ZERO
E1AF FD 2A 00 F0 0387 LD IY,(RAMTOP)
E1B3 01 92 FF 0388 LD BC,0-STORE;OFFSET
E1B6 FD 09 0389 ADD IY,BC ;SET UP IY
E1B8 C1 0390 POP BC
E1B9 C9 0391 RET
E1BA 0392 ;
0008 0393 WINPK5: EQU 8 ;DELAY FOR 15KC SIGNAL

```

```

E1BA      0395 ;
E1BA      0396 ;
E1BA      0397 ;
E1BA      0398 ;      UTILITY ROUTINES
E1BA      0399 ;
E1BA      0400 ;
E1BA      0401 ;
E1BA      0402 ; -----
E1BA      0403 ;
E1BA      0404 ;
E1BA      0405 ;
E1BA      0406 ;      MESSAGE OUTPUT ROUTINE
E1BA      0407 ;
E1BA      0408 ;      SCANS ASCII TEXT FOR:
E1BA      0409 ;          0 = RETURN
E1BA      0410 ;          CR = CRLF
E1BA      0411 ;      ALL OTHERS OUTPUT
E1BA      0412 ;
E1BA      0413 ;
E1BA      0414 MSGOUT: EQU    $
E1BA 7E      0415      LD    A,(HL)    ;GET CHR
E1BB B7      0416      OR     A          ;IS IT END?
E1BC C8      0417      RET    Z          ;YES- RETURN
E1BD 23      0418      INC    HL          ;NEXT
E1BE CD 45 E0 0419 MSGOT2: CALL  CHROUT    ;PRINT IT
E1C1 FE 0D      0420      CP     CR          ;NEED CRLF?
E1C3 20 F5      0421      JR     NZ,MSGOUT    ;NO
E1C5 3E 0A      0422      LD     A,LF        ;DO LF PART
E1C7 18 F5      0423      JR     MSGOT2
E1C9      0424 ;
E1C9      0425 ; -----
E1C9      0426 ;
E1C9      0427 ;
E1C9      0428 ;      "WHAT" ERROR ROUTINE
E1C9      0429 ;
E1C9      0430 ;
E1C9      0431 WHAT: EQU    $
E1C9 E5      0432      PUSH   HL          ;SAVE MSG ADDRESS
E1CA 21 DD E3   0433      LD     HL,ERRMSG  ;POINT "ERROR - "
E1CD CD BA E1   0434 WHAT1: CALL  MSGOUT
E1D0 E1      0435      POP     HL          ;GET BACK ERROR ADDRESS
E1D1 CD BA E1   0436      CALL  MSGOUT    ;PRINT IT
E1D4 FD 36 43 00 0437 FINISH: LD     (IY+BATCHF),0 ;CLEAR BATCH MODE
E1D8 CD B4 E2   0438      CALL  MTROF1    ;TURN OFF TAPE
E1DB C3 EB E0   0439      JP     START    ;REDO STACK
E1DE      0440 ;
E1DE      0441 ;      ERROR ROUTINES
E1DE      0442 ;
E1DE 21 F6 E3   0443 ERRPAR: LD     HL,IVPMMSG ;POINT "INVALID PARAMETER"
E1E1 18 E6      0444      JR     WHAT
E1E3 21 08 E4   0445 ERRCRC: LD     HL,CRCMSG ;POINT "TAPE CRC ERROR"
E1E6 18 E1      0446      JR     WHAT

```

```

E1E8      0448 ;
E1E8      0449 ;
E1E8      0450 ;
E1E8      0451 ;
E1E8      0452 ;
E1E8      0453 ;   HEXADECIMAL OUTPUT ROUTINES
E1E8      0454 ;
E1E8      0455 ;
E1E8      0456 ;
E1E8      0457 ;   ENTRIES:
E1E8      0458 ;
E1E8      0459 ;   ADDOUT = OUTPUT ADDRESS IN DE
E1E8      0460 ;   HCHOUT = OUTPUT BYTE IN A
E1E8      0461 ;
E1E8      0462 ;
E1E8      0463 ;
E1E8      0464 ADDOUT: EQU    $
E1E8      7A      0465      LD      A,D
E1E9      CD ED E1 0466      CALL   HCHOUT    ;PRINT MSB
E1EC      7B      0467      LD      A,E      ;PRINT LSB
E1ED      E1ED      0468 HCHOUT: EQU    $
E1ED      F5      0469      PUSH   AF        ;SAVE
E1EE      E6 F0    0470      AND     0F0H     ;ONLY LEFT HALF
E1F0      0F      0471      RRCA      ;MOVE RIGHT
E1F1      0F      0472      RRCA
E1F2      0F      0473      RRCA
E1F3      0F      0474      RRCA
E1F4      CD FA E1 0475      CALL   HCHOT2    ;FORM ASCII
E1F7      F1      0476      POP     AF        ;GET BACK CHAR
E1F8      E6 0F    0477      AND     0FH      ;ONLY RIGHT HALF
E1FA      FE 0A    0478 HCHOT2: CF     0AH      ;NEED LETTER?
E1FC      38 02    0479      JR      C,HCHOT3 ;NO
E1FE      C6 07    0480      ADD     'A'-3AH   ;ADJUST FOR A-F
E200      C6 30    0481 HCHOT3: ADD     30H      ;MAKE ASCII
E202      C3 45 E0 0482      JP      CHROUT   ;RETURN THERE

```

```

E205          0484 ;
E205          0485 ;
E205          0486 ;
E205          0487 ; CARRIAGE RETURN / LINE FEED
E205          0488 ;
E205          0489 ; ISSUES A <CR>,<LF> TO TERMINAL
E205          0490 ;
E205          0491 ;
E205          0492 ;
E205          0493 CRLF: EQU $
E205          0494          LD A,CR
E207          CD 45 E0      0495          CALL CHROUT
E20A          3E 0A        0496          LD A,LF
E20C          C3 45 E0      0497          JP CHROUT ;RETURN THERE
E20F          0498 ;
E20F          0499 ; -----
E20F          0500 ;
E20F          0501 ;
E20F          0502 ;
E20F          0503 ; ADDRESS AND COLON OUTPUT
E20F          0504 ;
E20F          0505 ; PRINTS ADDRESS IN DE FROM
E20F          0506 ; ADDOUT THEN PRINTS COLON
E20F          0507 ; AND A SPACE.
E20F          0508 ;
E20F          0509 ;
E20F          0510 ;
E20F          0511 ADDCOL: EQU $
E20F          CD E8 E1      0512          CALL ADDOUT ;PRINT ADDRESS
E212          3E 3A        0513          LD A,':' ;FORM COLON
E214          CD 45 E0      0514          CALL CHROUT ;AND SEND IT
E217          3E 20        0515          LD A, ' ' ;FORM SPACE
E219          C3 45 E0      0516          JP CHROUT ;SEND AND RETURN
E21C          0517 ;
E21C          0518 ; -----
E21C          0519 ;
E21C          0520 ;
E21C          0521 ;
E21C          0522 ; PRINT SPACE AND HEX BYTE
E21C          0523 ;
E21C          0524 ; PRINTS A SPACE AND THEN
E21C          0525 ; THE CHARACTER IN THE A
E21C          0526 ; REGISTER IN HEX.
E21C          0527 ;
E21C          0528 ;
E21C          0529 ;
E21C          0530 HEXSPC: EQU $
E21C          F5          0531          PUSH AF ;SAVE CHR.
E21D          3E 20        0532          LD A, ' ' ;FORM SPACE
E21F          CD 45 E0      0533          CALL CHROUT ;AND SEND IT
E222          F1          0534          POP AF ;GET CHAR BACK
E223          18 C8        0535          JR HCHOUT ;PRINT & RETURN

```



```

E225      0537 ;
E225      0538 ;
E225      0539 ;
E225      0540 ;   SCANNER ROUTINE
E225      0541 ;
E225      0542 ;
E225      0543 ;
E225      0544 ;   THIS ROUTINE SCANS THE
E225      0545 ;   INPUT BUFFER LOCATED IN
E225      0546 ;   THE STORAGE AREA AND SKIPS
E225      0547 ;   OVER EITHER:
E225      0548 ;           SCAN = DELIMITERS
E225      0549 ;           OR
E225      0550 ;           SCANLT = TEXT THEN SCAN
E225      0551 ;
E225      0552 ;   THIS ROUTINE USED FOR FINDING
E225      0553 ;   PARAMETERS IN I/O BUFFER
E225      0554 ;
E225      0555 ;
E225      0556 ;
E225      E225      0557 SCAN: EQU $
E225      7E      0558      LD      A,(HL)      ;GET ASCII
E226      FE 0D   0559      CP      CR          ;CAR RET?
E228      C8      0560      RET      Z          ;YES THRU
E229      FE 2E   0561      CP      ','        ;DELIM?
E22B      D0      0562      RET      NC        ;YES - GO BACK
E22C      23      0563      INC      HL        ;NEXT
E22D      18 F6   0564      JR      SCAN
E22F      0565 ;
E22F      0566 ;
E22F      E22F    0567 SCANHL: EQU $
E22F      FD E5   0568      PUSH    IY
E231      E1      0569      POP     HL          ;GET BUFF REG
E232      E232    0570 SCANLT: EQU $
E232      7E      0571      LD      A,(HL)      ;GET
E233      FE 0D   0572      CP      CR
E235      C8      0573      RET      Z          ;THRU IF CR
E236      FE 30   0574      CP      '0'        ;< 0?
E238      38 EB   0575      JR      C,SCAN      ;YES-GO UP
E23A      23      0576      INC      HL        ;NEXT
E23B      18 F5   0577      JR      SCANLT

```

```

E23D          0579 ;
E23D          0580 ;
E23D          0581 ;
E23D          0582 ;   CONVERSION ROUTINE
E23D          0583 ;
E23D          0584 ;
E23D          0585 ;   THIS ROUTINE SCANS THE ASCII
E23D          0586 ;   I/O BUFFER AND CONVERTS THE
E23D          0587 ;   ASCII HEX TEXT TO BINARY IN
E23D          0588 ;   THE DE REGISTER PAIR. VALUE
E23D          0589 ;   IS ROTATD IN THROUGH E, SO
E23D          0590 ;   IF ONLY ONE BYTE THEN USE E.
E23D          0591 ;
E23D          0592 ;   ERROR FOR INVALID ASCII ROUTES
E23D          0593 ;   TO WHAT ERROR ROUTINE.
E23D          0594 ;
E23D          0595 ;
E23D          0596 ;
E23D          E23D
E23D 11 00 00    0597 CONV:   EQU    $
E240 7E          0598       LD     DE,0       ;SET FOR NUMBER
E241 FE 30       0599 CONV1:  LD     A,(HL)    ;GET CHAR
E243 D8          0600       CP     '0'        ;DELIM?
E244 23          0601       RET     C         ;YES-THRU
E245 FE 47       0602       INC    HL         ;NEXT
E247 D2 DE E1    0603       CP     'F'+1      ;IS IT TOO BIG?
E24A FE 3A       0604       JP     NC,ERRPAR  ;YES
E24C 38 07       0605       CP     '9'+1      ;IS IT A #?
E24E FE 41       0606       JR     C,NUMBER   ;YES
E250 DA DE E1    0607       CP     'A'        ;IS IT A LETTER
E253 C6 09       0608       JP     C,ERRPAR   ;NO!
E255 07          0609       ADD    9         ;MAKE 10-15
E256 07          0610 NUMBER: RLCA          ;SHIFT
E257 07          0611       RLCA          ; TO
E258 07          0612       RLCA          ; LEFT
E259 06 04       0613       RLCA
E25B 07          0614       LD     B,4        ;COUNT
E25C CB 13       0615 CONV2:  RLCA          ;INTO CARRY
E25E CB 12       0616       RL     E         ;INTO E
E260 10 F9       0617       RL     D         ; AND D
E262 18 DC       0618       DJNZ  CONV2      ;TILL B=0
          0619       JR     CONV1          ;TRY AGAIN

```

```

E264      0621 ;
E264      0622 ;
E264      0623 ;
E264      0624 ; NAME FIND ROUTINE
E264      0625 ;
E264      0626 ;
E264      0627 ; THIS ROUTINE FINDS THE ASCII
E264      0628 ; NAME IN I/O BUFFER AND MOVES
E264      0629 ; IT TO CHEAD FILLING WITH SPACES
E264      0630 ; FOR 5 CHARACTERS.
E264      0631 ;
E264      0632 ; EXIT:      Z SET = NO NAME
E264      0633 ;           C SET = BAD NAME
E264      0634 ;
E264      0635 ;
E264      0636 NAMFND: EQU    $
E264      0637      CALL  SCANHL    ;SKIP COMMAND
E267      C8      0638      RET      Z      ;FLAG SET IF CR
E268      FE 41    0639      CP      'A'    ;IS IT
E26A      D8      0640      RET      C
E26A      6B FE 5B 0641      CP      'Z'+1    ;A LETTER?
E26A      6D 3F    0642      CCF          ;SET CARRY
E26E      D8      0643      RET      C      ; IF NOT
E26F      E5      0644      PUSH    HL      ;SAVE PNTR
E270      FD E5    0645      PUSH    IY      ;MOVE INDEX
E272      D1      0646      POP      DE      ; TO DE
E273      21 47 00 0647      LD      HL,CHEAD ;HL-OFFSET
E276      19      0648      ADD     HL,DE    ;HL-ADDRESS
E277      D1      0649      POP      DE      ;RESTORE PNTR
E278      06 05    0650      LD      B,5     ;5 CHRS
E27A      FE 30    0651 NAMEN1: CP      '0'    ;< 0?
E27C      13      0652      INC     DE      ;NEXT
E27D      30 03    0653      JR      NC,NAMEN2 ;NO-GO ON
E27F      1B      0654      DEC     DE      ;MOVE PNTR BACK
E280      3E 20    0655      LD      A,SPACE ;SPACE FILL
E282      77      0656 NAMEN2: LD      (HL),A ;PUT AWAY
E283      23      0657      INC     HL      ;POINT NEXT
E284      1A      0658      LD      A,(DE)   ;GET NEXT
E285      10 F3    0659      DJNZ   NAMEN1   ;GO FOR 5
E287      B7      0660      OR      A      ;REDO FLAGS
E288      EB      0661      EX      DE,HL    ;RESTORE HL
E289      C9      0662      RET

```

```

E28A      0664 ;
E28A      0665 ;
E28A      0666 ;
E28A      0667 ;    CASSETTE MOTOR CONTROL ROUTINES
E28A      0668 ;
E28A      0669 ;
E28A      0670 ;
E28A      E28A      0671 MOTRON: EQU    $
E28A      FD E5      0672          PUSH  IY
E28C      CD A2 E1    0673          CALL  GETIY
E28F      FD 7E 3D    0674          LD    A,(IY+TAPES);GET SPEED
E292      05          0675          DEC    B
E293      28 02       0676          JR     Z,MOTR01 ;NO
E295      C6 10       0677          ADD   10H
E297      C6 10       0678 MOTR01: ADD   10H
E299      D3 FE       0679          OUT   OFEH,A
E29B      FD 77 45    0680          LD    (IY+CMTRFG),A;PUT AWAY
E29E      FD E1       0681          POP   IY
E2A0      06 04       0682 DELAY:  LD    B,4          ;LOOP COUNT
E2A2      E5          0683 DELAY1: PUSH  HL          ;WE DESTROY
E2A3      21 00 00    0684 DELAY2: LD    HL,0          ;CLEAR IT
E2A6      2B          0685 DELAY3: DEC    HL
E2A7      7C          0686          LD    A,H
E2A8      B5          0687          OR    L
E2A9      20 FB       0688          JR     NZ,DELAY3 ;LOOP
E2AB      10 F6       0689          DJNZ  DELAY2    ;SOME MORE
E2AD      E1          0690          POP   HL          ;RESTORE
E2AE      C9          0691          RET           ;WE'RE THRU
E2AF      E2AF      0692 ;
E2AF      06 01       0693 MTROFF: EQU    $
E2B1      CD A2 E2    0694          LD    B,1
E2B4      FD E5       0695          CALL  DELAY1
E2B6      CD A2 E1    0696 MTROF1: PUSH  IY
E2B9      AF          0697          CALL  GETIY
E2BA      D3 FE       0698          XOR    A
E2BC      FD 77 45    0699          OUT   OFEH,A
E2BF      FD E1       0700          LD    (IY+CMTRFG),A;PUT AWAY
E2C1      C9          0701          POP   IY
E2C2      0702          RET           ;GO BACK
E2C2      0703 ;
E2C2      0704 ;    UART EQUATES
E2C2      0705 ;
E2C2      00FD       0706 UARTS:  EQU    OFDH
E2C2      00FC       0707 UARTD:  EQU    OFCH
E2C2      0708 ;
E2C2      0709 ;
E2C2      0710 ;    NULL ROUTINE
E2C2      0711 ;
E2C2      0712 ;
E2C2      E2C2      0713 NULL:    EQU    $
E2C2      06 64       0714          LD    B,100        ;SET B/#
E2C4      AF          0715 NULL1: XOR    A          ;FORM NULL
E2C5      CD EE E2    0716          CALL  TAPOUT      ;SEND IT
E2C8      10 FA       0717          DJNZ  NULL1        ;IS B 0?
E2CA      3C          0718          INC    A
E2CB      CD EE E2    0719          CALL  TAPOUT
E2CE      FD 70 46    0720          LD    (IY+CRCBYT),B;CLEAR CRC
E2D1      C9          0721          RET

```

```

E2D2          0722 ;
E2D2          0723 ;
E2D2          0724 ;      SPACES ROUTINE
E2D2          0725 ;
E2D2          0726 ;
E2D2          E2D2      0727 SPACES: EQU      $
E2D2          3E 20      0728          LD      A,SPACE
E2D4          CD 45 E0    0729          CALL   CHROUT
E2D7          10 F9      0730          DJNZ   SPACES      ;LOOP TINI B=0
E2D9          C9         0731          RET
E2DA          0732 ;
E2DA          0733 ;      CASSETTE TAPE INPUT / OUTPUT
E2DA          0734 ;
E2DA          0735 ;
E2DA          0736 ;      TAPE BYTE INPUT
E2DA          0737 ;
E2DA          E2DA      0738 TAPEIN: EQU      $
E2DA          FD E5      0739          PUSH   IY
E2DC          CD A2 E1    0740          CALL   GETIY      ;GO GET IY
E2DF          CD D1 EA    0741 TAPIN1: CALL   ESCCHK      ;USER?
E2E2          20 2B      0742          JR      NZ,TAPLVE ;HE WANTS US!
E2E4          DB FD      0743          IN      A,UARTS
E2E6          CB 4F      0744          BIT     1,A
E2E8          28 F5      0745          JR      Z,TAPIN1
E2EA          DB FC      0746          IN      A,UARTD
E2EC          18 0F      0747          JR      CRCOMP
E2EE          0748 ;
E2EE          0749 ;      TAPE BYTE OUTPUT
E2EE          0750 ;
E2EE          E2EE      0751 TAPOUT: EQU      $
E2EE          FD E5      0752          PUSH   IY
E2F0          CD A2 E1    0753          CALL   GETIY      ;GO GET IY
E2F3          F5         0754          PUSH   AF
E2F4          DB FD      0755 TAPOT1: IN      A,UARTS
E2F6          CB 47      0756          BIT     0,A
E2F8          28 FA      0757          JR      Z,TAPOT1
E2FA          F1         0758          POP    AF
E2FB          D3 FC      0759          OUT    UARTD,A
E2FD          0760 ;
E2FD          0761 ;      CRC COMPUTATION ROUTINE
E2FD          0762 ;
E2FD          C5         0763 CRCOMP: PUSH   BC      ;WE DESTROY
E2FE          F5         0764          PUSH   AF      ;ALSO
E2FF          FD 46 46    0765          LD      B,(IY+CRCBYT);GET CRC
E302          90         0766          SUB    B
E303          47         0767          LD      B,A
E304          A8         0768          XOR    B
E305          2F         0769          CPL
E306          90         0770          SUB    B
E307          FD 77 46    0771          LD      (IY+CRCBYT),A
E30A          F1         0772          POP    AF
E30B          C1         0773          POP    BC      ;RESTORE
E30C          FD E1      0774 TAPLV2: POP    IY      ;RESTORE
E30E          C9         0775          RET
E30F          0776 ;
E30F          0777 ;
E30F          AF        0778 TAPLVE: XOR    A
E310          18 FA      0779          JR      TAPLV2

```

```

E312      0780 ;
E312      0781 ;
E312      0782 ;
E312      0783 ;
E312      0784 ;      COMMAND TABLE
E312      0785 ;
E312      0786 ;      FORMATED AS FOLLOWS:
E312      0787 ;          2 BYTE ASCII COMMAND
E312      0788 ;          2 BYTE JUMP ADDRESS
E312      0789 ;          END BYTE IS 0
E312      0790 ;
E312      0791 ;
E312      0792 TABLE: EQU $
E312      44 55      0793      DB      'DU'
E314      D3 E4      0794      DW      DUMP      ;DUMP FROM MEMORY
E316      45 4E      0795      DB      'EN'
E318      38 E5      0796      DW      ENTER      ;ENTER TO MEMORY
E31A      53 41      0797      DB      'SA'
E31C      38 E6      0798      DW      SAVE      ;SAVE FILE ON CASSETTE
E31E      4C 4F      0799      DB      'LO'
E320      8A E7      0800      DW      LOAD      ;LOAD FILE FROM CASS.
E322      46 49      0801      DB      'FI'
E324      B9 E6      0802      DW      FILES      ;LIST CASSETTE FILES
E326      47 4F      0803      DB      'GO'
E328      97 E5      0804      DW      GO      ;GO TO PROGRAM
E32A      43 52      0805      DB      'CR'
E32C      5C E8      0806      DW      CREAT      ;CREAT BATCH FILE
E32E      53 45      0807      DB      'SE'
E330      A2 E5      0808      DW      SET      ;SET PARAMETERS
E332      4D 4F      0809      DB      'MO'
E334      62 E5      0810      DW      MOVE      ;MOVE BLOCK MEMORY
E336      54 45      0811      DB      'TE'
E338      A1 E8      0812      DW      TEST      ;TEST
E33A      42 41      0813      DB      'BA'
E33C      58 E8      0814      DW      BATCH      ;EXECUTE BATCH FILE
E33E      4C 49      0815      DB      'LI'
E340      84 E8      0816      DW      LIST      ;LIST BATCH FILE
E342      50 52      0817      DB      'PR'
E344      45 E8      0818      DW      PRMPTC      ;CHANGE PROMPT CHAR
E346      4F 56      0819      DB      'OV'
E348      D4 E1      0820      DW      FINISH      ;END BATCH MODE
E34A      50 50      0821      DB      'PP'
E34C      8A E9      0822      DW      PROMPK      ;BRANCH TO FROM PACK
E34E      00      0823      ENDTBL: EQU $
E34E      00      0824      DB      0

```

```

E34F      0826 ;
E34F      0827 ;
E34F      0828 ;
E34F      0829 ;
E34F      0830 ;
E34F      0831 ;      SET COMMAND TABLE
E34F      0832 ;
E34F      0833 ;
E34F      0834 ;
  34F      0835 ;
  34F      0836 ;
    E34F      0837 SETTBL: EQU    $
  34F      0838 ;
  34F  54      0839      DB      'T'
  350 DE E5      0840      DW      TAPE      ;SET TAPE RATE
  352 53      0841      DB      'S'
  353 EA E5      0842      DW      SPEED      ;SET DISPLAY SPEED
  355 58      0843      DB      'X'
  356 F2 E5      0844      DW      XEQSET      ;SET XEQ ADDRESS
  E358 46      0845      DB      'F'
  E359 EE E5      0846      DW      SETFIL      ;SET FILE TYPE
  E35B 4F      0847      DB      'O'
  E35C F9 E5      0848      DW      SETOUT      ;SET OUTPUT
  E35E 49      0849      DB      'I'
  E35F 1C E6      0850      DW      SETIN      ;SET INPUT
  E361 00      0851      DB      0

```

E362		0853 ;		
E362		0854 ;		
E362		0855 ;		
E362		0856 ;		
E362		0857 ;	MESSAGE TABLE	
E362		0858 ;		
E362		0859 ;		
E362		0860 ;		
E362		0861 ;		
E362	0D	0862	HEDING: DB	CR
E363	45 58 49 44	0863	DB	'EXIDY STANDARD MONITOR'
	59 20 53 54			
	41 4E 44 41			
	52 44 20 4D			
	4F 4E 49 54			
	4F 52			
E379	0D 0D	0864	DB	CR,CR
E37B	56 45 52 53	0865	DB	'VERSION 1.0'
	49 4F 4E 20			
	31 2E 30			
E386	0D	0866	DB	CR
E387	43 4F 50 59	0867	DB	'COPYRIGHT (C) 1978 BY '
	52 49 47 48			
	54 20 28 43			
	29 20 31 39			
	37 38 20 42			
	59 20			
E39D	45 58 49 44	0868	DB	'EXIDY INC.'
	59 20 49 4E			
	43 2E			
E3A7	0D 0D	0869	DB	CR,CR
E3A9	54 48 45 20	0870	DB	'THE TOP OF RAM IS '
	54 4F 50 20			
	4F 46 20 52			
	41 4D 20 49			
	53 20			
E3BB	00	0871	DB	0
E3BC	20 48 45 58	0872	DB	'HEX.'
	2E			
E3C1	0D	0873	DB	CR
E3C2	53 54 41 43	0874	DB	'STACK BEGINS FROM ',0
	4B 20 42 45			
	47 49 4E 53			
	20 46 52 4F			
	4D 20 00			
E3D5	20 48 45 58	0875	DB	'HEX.'
	2E			
E3DA	0D 0D 00	0876	DB	CR,CR,0
E3DD		0877 ;		
E3DD	45 52 52 4F	0878	DB	'ERROR - ',0
	52 20 2D 20			
	00			
E3E6	49 4E 56 41	0879	DB	'INVALID COMMAND',0
	4C 49 44 20			
	43 4F 4D 4D			
	41 4E 44 00			
E3F6	49 4E 56 41	0880	DB	'INVALID PARAMETER',0
	4C 49 44 20			



	50 41 52 41			
	40 45 54 45			
	52 00			
E408	54 41 50 45	0881	CRCMSG: DB	'TAPE CRC ERROR',0
	20 43 52 43			
	20 45 52 52			
	4F 52 00			
E417		0882 ;		
E417	0D	0883 DHEAD: DB	CR	
E418	41 44 44 52	0884	DB	'ADDR 0 1 2 3'
	20 20 20 30			
	20 20 31 20			
	20 32 20 20			
	33			
E429	20 20 20 34	0885	DB	' 4 5 6 7'
	20 20 35 20			
	20 36 20 20			
	37			
E436	20 20 20 38	0886	DB	' 8 9 A B'
	20 20 39 20			
	20 41 20 20			
	42			
E443	20 20 20 43	0887	DB	' C D E F'
	20 20 44 20			
	20 45 20 20			
	46			
E450	0D 0D 00	0888	DB	CR,CR,0
E453		0889 ;		
E453	0D 0D	0890 FILHD: DB	CR,CR	
E455	4E 41 4D 45	0891	DB	'NAME
	20 20 20			
E45C	46 49 4C 45	0892	DB	'FILE
	20			
E461	42 4C 43 4B	0893	DB	'BLCK ADDR
	20 41 44 44			
	52 20			
E46B	47 4F 41 44	0894	DB	'GOADDRS'
	44 52 53			
E472	0D 0D 00	0895	DB	CR,CR,0
E475		0896 ;		
E475		0897 ;		
E475	0D 41 44 44	0898 TESTHD: DB	CR,'ADDR	BIT'
	52 20 20 20			
	42 49 54			
E480	20 30 20 20	0899	DB	' 0 1
	20 31 20 20			
	20			
E489	32 20 20 20	0900	DB	'2 3
	33 20 20 20			
E491	34 20 20 20	0901	DB	'4 5
	35 20 20 20			
E499	36 20 20 20	0902	DB	'6 7',CR,CR,0
	37 0D 0D 00			
E4A1	42 41 44 20	0903 BADMSG: DB	'BAD	',0
	00			
E4A6	4F 4B 20 20	0904 OKMSG: DB	'OK	',0
	00			
E4AB	20 20 50 41	0905 PSCMSG: DB	' PASS COMPLETED.'	,CR,CR,0

	53	53	20	43		
	4F	4D	50	4C		
	45	54	45	44		
	2E	0D	0D	00		
E4BF					0906 ;	
E4BF	0D	4C	4F	41	0907 LDGMSG: DB	CR,'LOADING -',0
	44	49	4E	47		
	20	2D	00			
E4CA	46	4F	55	4E	0908 FNDMSG: DB	'FOUND - ',0
	44	20	2D	20		
	00					

# EXIDY STANDARD MONITOR SOFTWARE

```

E538      0967 ;
E538      0968 ;
E538      0969 ;
E538      0970 ;      ENTER COMMAND
E538      0971 ;
E538      0972 ;
E538      0973 ;
E538      0974 ENTER: EQU      $
E538      0975      CALL      CRLF      ;NEXT LINE
E538      0976      CALL      SCANHL    ;SKIP "EN"
E538      0977      JP        Z,ERRPAR  ;EOL?
E538      0978      CALL      CONV      ;GET ADDRESS
E538      0979 ENTER1: CALL      ADDCOL   ;PRINT ADDRESS
E538      0980      PUSH      DE        ;SAVE IT
E538      0981      CALL      LINEIN    ;GET A LINE
E538      0982      PUSH      IY        ;GET BUFFER
E538      0983      POP       HL        ; INTO HL
E538      0984      POP       DE        ; AND ADDRESS
E538      0985 ENTER2: CALL      SCAN    ;FIND PARAM
E538      0986      JP        Z,ENTER1  ;CR-LOOP
E538      0987      CP        '/'      ;END?
E538      0988      RET        Z        ;YES
E538      0989      PUSH      DE        ;SAVE-CONV DESTROYS
E538      0990      CALL      CONV      ;MAKE A #
E538      0991      LD        A,E      ; IN A
E538      0992      POP       DE        ;RESTORE
E538      0993      LD        DE, (DE),A ; AND MEM
E538      0994      INC       DE        ;NEXT
E538      0995      JR        ENTER2   ;AGAIN
E538      CD 05 E2
E538      CD 2F E2
E538      CA DE E1
E538      CD 3D E2
E538      CD 0F E2
E538      D5
E538      CD 3A E1
E538      FD E5
E538      E1
E538      D1
E538      CD 25 E2
E538      CA 44 E5
E538      FE 2F
E538      C8
E538      D5
E538      CD 3D E2
E538      7B
E538      D1
E538      12
E538      13
E538      18 ED

```

```

E562      0997 ;
E562      0998 ;
E562      0999 ;
E562      1000 ;      MOVE BLOCK ROUTINE
E562      1001 ;
E562      1002 ;
E562      1003 ;
E562      1004 MOVE:   EQU      $
E562      1005      CALL    SCANHL      ;SKIP "MO"
E562      1006      JP      Z,ERRPAR
E562      1007      CALL    CONV
E562      1008      PUSH    DE          ;SAVE "FROM"
E562      1009      CALL    SCAN
E562      1010      JP      Z,ERRPAR
E562      1011      CALL    CONV
E562      1012      PUSH    DE          ;SAVE "FROM END"
E562      1013      CALL    SCAN
E562      1014      CP      'S'        ;SWATH?
E562      1015      JR      Z,MOVE2     ;YES-ALL SET!
E562      1016      CALL    CONV       ;GET SWATH
E562      1017      SCF
E562      1018      CCF              ;CLEAR CARRY
E562      1019      POP     HL         ;GET "FROM END"
E562      1020      POP     BC         ;GET "FROM BEG"
E562      1021      PUSH    BC         ;SAVE AGAIN
E562      1022      SBC     HL,BC      ;MAKE SWATH
E562      1023      PUSH    HL         ;MOVE HL
E562      1024      POP     BC         ; TO BC
E562      1025 MOVE1:  POP     HL         ;GET "FROM"
E562      1026      INC     BC         ;ADJUST
E562      1027      LDIR
E562      1028      RET
E562      1029 MOVE2:  INC     HL         ;SKIP "S"
E562      1030      CALL    CONV       ;GET SWATH
E562      1031      PUSH    DE         ;MOVE DE
E562      1032      POP     BC         ; TO BC
E562      1033      POP     DE         ;GET "TO"
E562      1034      JR      MOVE1      ;CONTINUE UPSTAIRS
E562      18 F2

```

E597		1036 ;			
E597		1037 ;			
E597		1038 ;			
E597		1039 ;	GO COMMAND		
E597		1040 ;			
E597		1041 ;			
E597		1042 ;			
	E597	1043 GO:	EQU	\$	
E597	CD 2F E2	1044	CALL	SCANHL	;SKIP "GO"
E59A	CA DE E1	1045	JP	Z,ERRPAR	
E59D	CD 3D E2	1046	CALL	CONV	;GET ADDRESS
E5A0	EB	1047	EX	DE,HL	;PUT IN HL
E5A1	E9	1048	JP	(HL)	;GO TO IT!

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

```

E5A2      1050 ;
E5A2      1051 ;
E5A2      1052 ;
E5A2      1053 ;      SET COMMAND
E5A2      1054 ;
E5A2      1055 ;
E5A2      1056 ;
E5A2      1057 SET:   EQU    $
E5A2      1058      CALL  SCANHL    ;SKIP "SE"
E5A5      CD 2F E2      JP      Z,ERRPAR
E5A8      CA DE E1      LD      IX,SETTBL ;POINT SET TABLE
E5AC      DD 21 4F E3   1060      CP      (IX)      ;IS IT?
E5AF      DD BE 00      1061 SET1:   JR      Z,SET2    ;YES-GO SET UP
E5B1      DD 23          1062      INC     IX          ;NO-
E5B3      DD 23          1063      INC     IX          ; POINT
E5B5      DD 23          1064      INC     IX          ; NEXT
E5B7      F5            1065      INC     IX          ;
E5B8      DD 7E 00      1066      PUSH   AF          ;SAVE CHAR
E5BB      B7            1067      LD      A,(IX)      ;IS IT
E5BC      CA DE E1      1068      OR      A          ; END?
E5BE      F1            1069      JP      Z,ERRPAR    ;YES-INVALID
E5C0      18 EA          1070      POP     AF          ;RESTORE
E5C2      23            1071      JR      SET1        ;CONTINUE
E5C3      CD 25 E2      1072 SET2:   INC     HL          ;SKIP CHAR
E5C6      FE 3D          1073      CALL   SCAN        ;TO NEXT
E5C8      C2 DE E1      1074      CP      '='        ;?
E5CB      23            1075      JP      NZ,ERRPAR    ;NO-INVALID
E5CC      CD 25 E2      1076      INC     HL          ;SKIP "="
E5CF      CA DE E1      1077      CALL   SCAN        ;AND DELIMS
E5D2      FE 47          1078      JP      Z,ERRPAR    ;BAD END
E5D4      30 03          1079      CP      'G'
E5D6      CD 3D E2      1080      JR      NC,SET3
E5D9      E5D9          1081      CALL   CONV        ;MAKE FOR SET
E5DB      C3 1C E1      1082 SET3:   EQU     $
E5D9      DD 23          1083      INC     IX          ;FOR MAIN3
E5DB      C3 1C E1      1084      JP      MAIN3      ;JUMP FROM TBL

```

```

E5DE      1086 ;
E5DE      1087 ;
E5DE      1088 ;      SET VALUE ROUTINES
E5DE      1089 ;
E5DE      1090 ;
E5DE      1091 ;
E5DE      1092 ;      SET TAPE RATE
E5DE      1093 ;
      E5DE      1094 TAPE: EQU $
E5DE      7B      1095 LD A,E
E5DF      B7      1096 OR A ;TEST IF ZERO
E5E0      3E 00    1097 LD A,0 ;SET IN CASE 300 BAUD
E5E2      20 02    1098 JR NZ,TAPE1 ;GO DO 300 BAUD
E5E4      3E 40    1099 LD A,40H ;MAKE 1200 BAUD
E5E6      FD 77 3D 1100 TAPE1: LD (IY+TAPES),A;PUT AWAY
E5E9      C9      1101 RET
E5EA      1102 ;
E5EA      1103 ;      SET DISPLAY SPEED
E5EA      1104 ;
      E5EA      1105 SPEED: EQU $
E5EA      FD 73 3E 1106 LD (IY+SPEEDS),E
E5ED      C9      1107 RET
E5EE      1108 ;
E5EE      1109 ;      SET CASSETTE FILE TYPE
E5EE      1110 ;
      E5EE      1111 SETFIL: EQU $
E5EE      FD 73 4D 1112 LD (IY+CHEAD+HTYPE),E
E5F1      C9      1113 RET
E5F2      1114 ;
E5F2      1115 ;      SET XEQ ADDRESS
E5F2      1116 ;
      E5F2      1117 XEQSET: EQU $
E5F2      FD 73 52 1118 LD (IY+CHEAD+HXEQ),E
E5F5      FD 72 53 1119 LD (IY+CHEAD+HXEQ+1),D
E5F8      C9      1120 RET
E5F9      1121 ;
E5F9      1122 ;      SET OUTPUT ADDRESS
E5F9      1123 ;
E5F9      FE 56    1124 SETOUT: CP 'V'
E5FB      20 03    1125 JR NZ,SETOT1
E5FD      11 1B E0 1126 LD DE,VIDEO E01B 27,229
E600      FE 50    1127 SETOT1: CP 'P'
E602      20 03    1128 JR NZ,SETOT2
E604      11 21 E0 1129 LD DE,PARLOT E021 33,229
E607      FE 53    1130 SETOT2: CP 'S'
E609      20 03    1131 JR NZ,SETOT3
E60B      11 12 E0 1132 LD DE,OUTAPE E012 18,229
E60E      FE 4C    1133 SETOT3: CP 'L' ;IS IT CENTRONICS
E610      20 03    1134 JR NZ,SETOT4 ;NO
E612      11 93 E9 1135 LD DE,CENDRV E993 147,233
E615      FD 73 3F 1136 SETOT4: LD (IY+OUTADD),E
E618      FD 72 40 1137 LD (IY+OUTADD+1),D
E61B      C9      1138 RET
E61C      1139 ;
E61C      1140 ;      SET INPUT ADDRESS
E61C      1141 ;
      E61C      1142 SETIN: EQU $
E61C      FE 4B    1143 CP 'K'

```

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

E61E	20	03		1144	JR	NZ,SETIN1		
E620	11	18	E0	1145	LD	DE,KEYBRD	E018	24,224
E623	FE	50		1146	SETIN1: CP	'P'		
E625	20	03		1147	JR	NZ,SETIN2		
E627	11	1E	E0	1148	LD	DE,PARLIN	E01E	30,224
E62A	FE	53		1149	SETIN2: CP	'S'		
E62C	20	03		1150	JR	NZ,SETIN3		
E62E	11	0F	E0	1151	LD	DE,INTAPE	E01F	15,224
E631	FD	73	41	1152	SETIN3: LD	(IY+INADD),E		
E634	FD	72	42	1153	LD	(IY+INADD+1),D		
E637	C9			1154	RET			



```

E638      1156 ;
E638      1157 ;
E638      1158 ;
E638      1159 ;      SAVE COMMAND
E638      1160 ;
E638      1161 ;
E638      1162 ;
E638      1163 SAVE: EQU $
E638      1164 CALL NAMFND ;GET NAME
E638      1165 JP Z,ERRPAR ;NO NAME
E638      1166 JP C,ERRPAR ;BAD NAME
E638      1167 CALL SCANLT ;SKIP NAME
E638      1168 JP Z,ERRPAR ;EOL
E638      1169 CALL CONV ;GET BEG ADD
E638      1170 PUSH DE ;SAVE
E638      1171 LD (IY+CHEAD+HADDR),E
E638      1172 LD (IY+CHEAD+HADDR+1),D
E638      1173 CALL SCAN ;NEXT
E638      1174 JP Z,ERRPAR ;NO END ADD
E638      1175 CALL CONV
E638      1176 SAVBAS: EX DE,HL ;SAVE HL
E638      1177 POP BC ;GET BEG
E638      1178 PUSH BC ;RESAVE
E638      1179 SCF ;CLEAR ARRY
E638      1180 CCF
E638      1181 SBC HL,BC
E638      1182 INC HL ;ADJUST
E638      1183 PUSH HL ;SAVE BLK
E638      1184 LD (IY+CHEAD+HSIZE),L
E638      1185 LD (IY+CHEAD+HSIZE+1),H
E638      1186 LD (IY+CHEAD+5),55H ;MAKE AN EXIDY FILE
E638      1187 EX DE,HL
E638      1188 LD B,1 ;DEFAULT
E638      1189 CALL SCAN ;SKIP TO EOL
E638      1190 JR Z,SAVE1
E638      1191 CALL CONV ;GET UNIT
E638      1192 LD B,E ; IN B
E638      1193 SAVE1: CALL MOTRON ;TURN ON CAS.
E638      1194 CALL NULL ;& NULL IT
E638      1195 PUSH IY ;MOVE IY
E638      1196 POP IX ; TO IX
E638      1197 LD B,HEADLN ;HEADER LENGTH
E638      1198 SAVE2: LD A,(IX+CHEAD) ;GET
E638      1199 CALL TAPOUT ;SEND
E638      1200 INC IX ;NEXT
E638      1201 DJNZ SAVE2 ;LOOP FOR HEADER
E638      1202 CALL WRCRC ;WRITE CRC
E638      1203 CALL NULL ;WRITE NULLS AFTER HEADER
E638      1204 POP DE ;GET BLOCK SIZE
E638      1205 POP HL ;GET BEGGINNING ADDRESS
E638      1206 SAVE3: CALL BLKADJ ;DO BLOCK ADJUST
E638      1207 JP Z,MTROFF ;GO TURN OFF IF THRU
E638      1208 SAVE4: LD A,(HL) ;GET BYTE
E638      1209 CALL TAPOUT ;SEND IT
E638      1210 INC HL ;NEXT
E638      1211 DJNZ SAVE4 ;LOOP FOR BLOCK
E638      1212 CALL WRCRC ;WRITE CRC
E638      1213 JR SAVE3 ;KEEP GOIN'

```

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

E6A9		1214 ;		
E6A9		1215 ;	BLOCK ADJUST ROUTIN	
E6A9		1216 ;		
	E6A9	1217	BLKADJ: EQU	\$
E6A9	AF	1218	XOR	A ;CLEAR A
E6AA	FD 77 46	1219	LD	(IY+CRCBYT),A;CLEAR CRC
E6AD	47	1220	LD	B,A ;ALSO B
E6AE	B2	1221	OR	D ;IS D ZERO?
E6AF	20 05	1222	JR	NZ,BLKAJ2 ;NO-NORMAL RETURN
E6B1	B3	1223	OR	E ;IS E ZERO?
E6B2	C8	1224	RET	Z ;YES-WE'RE THRU!
E6B3	43	1225	LD	B,E ;DO ADJUSTMENT ON DE
E6B4	5A	1226	LD	E,D
E6B5	C9	1227	RET	
E6B6	15	1228	BLKAJ2: DEC	D ;ONE LESS BLOCK
E6B7	B7	1229	OR	A ;RESET ZERO FLAG
E6B8	C9	1230	RET	;BYE BYE

```

E6B9          1232 ;
E6B9          1233 ;
E6B9          1234 ;   FILES COMMAND
E6B9          1235 ;
E6B9          1236 ;   LISTS FILES FROM CASSETTE TO
E6B9          1237 ;   TERMINAL.
E6B9          1238 ;
E6B9          1239 FILES: EQU   $
E6B9  CD 2F E2 1240          CALL  SCANHL      ;SKIP "FI"
E6BC  06 01    1241          LD      B,1        ;UNIT DEFAULT
E6BE  28 04    1242          JR      Z,FILES1
E6C0  CD 3D E2 1243          CALL  CONV        ;GET UNIT
E6C3  43      1244          LD      B,E        ; IN B
E6C4  21 53 E4 1245 FILES1: LD      HL,FILHD
E6C7  CD BA E1 1246          CALL  MSGOUT      ;SEND HEADING
E6CA  CD 8A E2 1247          CALL  MOTRON     ;TURN ON!
E6CD  CD 1B E7 1248 FILES2: CALL  GETHED
E6D0  CD DE E6 1249          CALL  HEDPRT     ;PRINT HEADER
E6D3  FD 7E 5C 1250          LD      A,(IY+THEAD+5);GET EXIDY FILE CHECK
E6D6  B7      1251          OR      A        ;SET FLAGS
E6D7  28 F4    1252          JR      Z,FILES2   ;DO PROC TECH SKIP
E6D9  CD 34 E7 1253          CALL  SKIPFL    ;NEXT FILE
E6DC  18 EF    1254          JR      FILES2
E6DE          1255 ;
E6DE          1256 ;-----
E6DE          1257 ;
E6DE          1258 ;   PRINTS HEADER ON TERMINAL
E6DE          1259 ;
E6DE          1260 ;
E6DE  FD E5    1261 HEDPRT: PUSH  IY
E6E0  DD E1    1262          POP   IX        ;MOVE IY>IX
E6E2  06 05    1263          LD      B,5        ;NAME
E6E4  DD 7E 57 1264 FILES3: LD      A,(IX+THEAD);GET
E6E7  CD 45 E0 1265          CALL  CHROUT    ;SEND
E6EA  DD 23    1266          INC     IX        ;NEXT
E6EC  10 F6    1267          DJNZ   FILES3    ;LOOP FOR 5
E6EE  06 03    1268          LD      B,3
E6F0  CD D2 E2 1269          CALL  SPACES    ;3 SPACES
E6F3  DD 23    1270          INC     IX        ;SKIP OVER ZERO IN HEADER
E6F5  DD 7E 57 1271          LD      A,(IX+THEAD)
E6F8  CD 45 E0 1272          CALL  CHROUT    ;SEND FILE TYPE
E6FB  06 03    1273          LD      B,3
E6FD  CD D2 E2 1274          CALL  SPACES    ;3 SPACES
E700  DD 23    1275          INC     IX        ;NEXT
E702  06 03    1276          LD      B,3        ;THREE ADDRESS
E704  DD 5E 57 1277 FILES4: LD      E,(IX+THEAD)
E707  DD 56 58 1278          LD      D,(IX+THEAD+1);GET ADD
E70A  DD 23    1279          INC     IX
E70C  DD 23    1280          INC     IX        ;SKIP THIS 1
E70E  CD E8 E1 1281          CALL  ADROUT    ;PRINT IT
E711  3E 20    1282          LD      A,SPACE
E713  CD 45 E0 1283          CALL  CHROUT    ;1 SPACE
E716  10 EC    1284          DJNZ   FILES4    ;LOOP FOR 3
E718  C3 05 E2 1285          JP      CRLF     ;NEW LINE

```

```

E71B      1287 ;
E71B      1288 ;
E71B      1289 ;
E71B      1290 ;      CASSETTE UTILITY ROUTINES
E71B      1291 ;
E71B      1292 ;
E71B      1293 ; -----
E71B      1294 ;
E71B      1295 ;      GET HEADER
E71B      1296 ;
E71B      1297 ;      LOADS HEADER FROM TAPE TO BUFFER
E71B      1298 ;
E71B      1299 GETHED: EQU      $
E71B      1300      CALL      TAPWT      ;SKIP NULLS
E71E      1301      PUSH      IY      ;MOVE IY
E720      1302      POP       IX      ; TO IX
E722      1303      LD        B,HEADLN ;GET LENGTH
E724      1304 GETHD1: CALL      TAPEIN ;GET BYTE
E727      1305      JP        Z,FINISH ;USER WANTS US
E72A      1306      LD        (IX+THEAD),A ;MOVE IT
E72D      1307      INC       IX      ;NEXT
E72F      1308      DJNZ      GETHD1    ;LOOP
E731      1309      JP        CKCRC    ;CHECK CRC & RETURN
E734      1310 ;
E734      1311 ;
E734      1312 ; -----
E734      1313 ;
E734      1314 ;      SKIP A CASSETTE FILE
E734      1315 ;
E734      1316 ;      THIS ROUTINE SKIPS A FILE ON TAPE
E734      1317 ;      WITHOUT LOADING IT INTO MEMORY.
E734      1318 ;
E734      1319 ;
E734      1320 SKIPFL: EQU      $
E734      1321      CALL      TAPWT      ;WAIT FOR THE NULLS
E737      1322      LD        E,(IY+THEAD+HSIZE);GET BLK SIZE
E73A      1323      LD        D,(IY+THEAD+HSIZE+1)
E73D      1324 SKIPF1: CALL      BLKADJ    ;GO ADJUST BLOCK
E740      1325      RET        Z      ;THAT ' S ALL!
E741      1326 SKIPF2: CALL      TAPEIN    ;GET FROM TAPE
E744      1327      JP        Z,FINISH
E747      1328      DJNZ      SKIPF2    ;FOR ENTIRE BLOCK
E749      1329      CALL      CKCRC    ;CHECK CRC
E74C      1330      JR        SKIPF1    ;MORE-
E74E      1331 ;
E74E      1332 ;      CHECK CRC ON TAPE
E74E      1333 ;
E74E      1334 CKCRC: EQU      $
E74E      1335      LD        B,(IY+CRCBYT)
E751      1336      CALL      TAPEIN
E754      1337      CF        B
E755      1338      JP        NZ,ERRCRC
E758      1339      RET

```

```

E759      1341 ;
E759      1342 ;
E759      1343 ;
E759      1344 ;      TAPE WAIT ROUTINE
E759      1345 ;
E759      1346 ;      THIS ROUTINE WAITS FOR TEN NULLS
E759      1347 ;      FOLLOWED BY OTHER NULLS TILL A 1
E759      1348 ;
      E759      1349 TAPWT: EQU      $
E759      C5      1350      PUSH    BC          ;WE DESTROY
E75A      06 0A    1351 TAPWT1: LD      B,10
E75C      CD DA E2 1352 TAPWT2: CALL   TAPEIN
E75F      CA D4 E1 1353      JP      Z,FINISH
E762      B7      1354      OR      A          ;IS IT A NULL?
E763      20 F5    1355      JR      NZ,TAPWT1 ;NO-START OVER
E765      10 F5    1356      DJNZ   TAPWT2      ;LOOP FOR 10
E767      CD DA E2 1357 TAPWT3: CALL   TAPEIN
E76A      CA D4 E1 1358      JP      Z,FINISH
E76D      FE 01    1359      CP      1          ;A ONE?
E76F      20 F6    1360      JR      NZ,TAPWT3
E771      FD 70 46 1361      LD      (IY+CRCBYT),B;CLEAR CRC
E774      C1      1362      POP     BC
E775      C9      1363      RET
E776      1364 ;
E776      1365 ;-----
E776      1366 ;
E776      1367 ;
E776      1368 ;      PARALLEL I/O ROUTINES
E776      1369 ;
E776      1370 ;
      E776      1371 PARIN: EQU      $
E776      DB FE    1372      IN      A,0FEH
E778      CB 7F    1373      BIT     7,A
E77A      28 FA    1374      JR      Z,PARIN
E77C      DB FF    1375      IN      A,0FFH
E77E      C9      1376      RET
E77F      1377 ;
      E77F      1378 PAROUT: EQU     $
E77F      F5      1379      PUSH   AF
E780      DB FE    1380 PAROT1: IN      A,0FEH
E782      CB 77    1381      BIT     6,A
E784      28 FA    1382      JR      Z,PAROT1
E786      F1      1383      POP     AF
E787      D3 FF    1384      OUT     OFFH,A
E789      C9      1385      RET

```

```

E78A      1387 ;
E78A      1388 ;
E78A      1389 ;
E78A      1390 ;      LOAD COMMAND
E78A      1391 ;
E78A      1392 ;
E78A      1393 ;
E78A      1394 LOAD: EQU $
E78A      1395      CALL SCANHL      ;SKIP "L"
E78D      2B      1396 LOAD1: DEC HL      ;CHK FOR "G"
E78E      7E      1397      LD A,(HL)
E78F      FE 30    1398      CP '0'
E791      38 FA    1399      JR C,LOAD1      ;SKIP DELIMS
E793      FE 47    1400      CP 'G'      ;IS IT A "G"
E795      F5      1401      PUSH AF      ;SAVE TEST FLGS
E796      CD 64 E2  1402      CALL NAMFND      ;GET NAME
E799      F5      1403 LODBAS: PUSH AF      ;SAVE ALSO
E79A      06 01    1404      LD B,1      ;DEFAULT UNIT
E79C      F5      1405      PUSH AF      ;FOR LATER
E79D      28 19    1406      JR Z,LOAD3      ;GO LOAD
E79F      F1      1407      POP AF      ;DON'T NEED
E7A0      38 07    1408      JR C,LOAD2
E7A2      CD 32 E2  1409      CALL SCANLT      ;SKIP NAME
E7A5      F5      1410      PUSH AF
E7A6      28 10    1411      JR Z,LOAD3      ;GO LOAD
E7A8      F1      1412      POP AF
E7A9      CD 3D E2  1413 LOAD2: CALL CONV      ;MAKE UNIT
E7AC      43      1414      LD B,E      ; IN B
E7AD      CD 25 E2  1415      CALL SCAN      ;SKIP OVER
E7B0      F5      1416      PUSH AF      ;SAVE FLAGS
E7B1      28 05    1417      JR Z,LOAD3      ;GO LOAD IF EOL
E7B3      C5      1418      PUSH BC
E7B4      CD 3D E2  1419      CALL CONV
E7B7      C1      1420      POP BC      ;RESTORE
E7B8      CD 05 E2  1421 LOAD3: CALL CRLF      ;START WITH FRESH LINE
E7BB      CD 8A E2  1422      CALL MOTRON      ;WHAT A TURN ON!
E7BE      D5      1423 LOAD3A: PUSH DE      ;LOAD ADDRESS
E7BF      CD 1B E7  1424      CALL GETHED      ;GET HEADER
E7C2      FD 7E 5C  1425      LD A,(IY+THEAD+5);GET EXIDY FILE CHECK
E7C5      B7      1426      OR A      ;SET FLAGS
E7C6      28 0B    1427      JR Z,LOAD3B      ;NO PRINTING FOR PT
E7C8      E5      1428      PUSH HL      ;WE NEED RIGHT NOW!
E7C9      21 CA E4  1429      LD HL,FNDMSG      ;POINT TO "FOUND -"
E7CC      CD BA E1  1430      CALL MSGOUT      ;PRINT IT
E7CF      CD DE E6  1431      CALL HEDPRT      ;PRINT TAPE HEADER
E7D2      E1      1432      POP HL      ;GET BACK
E7D3      D1      1433 LOAD3B: POP DE      ;RESTORE ADD
E7D4      F1      1434      POP AF      ;FLAGS
E7D5      F5      1435      PUSH AF
E7D6      20 06    1436      JR NZ,LOAD5      ;ADD IN HEADER
E7D8      FD 5E 60  1437      LD E,(IY+THEAD+HADDR);GET ADD
E7DB      FD 56 61  1438      LD D,(IY+THEAD+HADDR+1)
E7DE      E1      1439 LOAD5: POP HL
E7DF      F1      1440      POP AF      ;NAME?
E7E0      F5      1441      PUSH AF      ;PUT BACK
E7E1      E5      1442      PUSH HL      ;DITTO
E7E2      28 14    1443      JR Z,LOAD7      ;NOPE-GO LOAD
E7E4      38 12    1444      JR C,LOAD7

```

E7E6	FD E5	1445	PUSH	IY	;PUT IY
E7E8	DD E1	1446	POP	IX	; IN IX
E7EA	06 05	1447	LD	B,5	;NAME LGTH
E7EC	DD 7E 47	1448	LOAD6:	LD	A,(IX+CHEAD);GET
E7EF	DD BE 57	1449	CP	(IX+THEAD);SAME?	
E7F2	DD 23	1450	INC	IX	;NEXT
E7F4	20 49	1451	JR	NZ,LOADSK	;GO SKIP
E7F6	10 F4	1452	DJNZ	LOAD6	;KEEP GOIN
E7F8	FD 7E 5C	1453	LOAD7:	LD	A,(IY+THEAD+5);GET EXIDY FILE CH
E7FB	B7	1454	OR	A	;SET FLAGS
E7FC	28 09	1455	JR	Z,LOAD7A	;NO PRINTING FOR PT
E7FE	21 BF E4	1456	LD	HL,LDMMSG	;POINT TO "LOADING -"
E801	CD BA E1	1457	CALL	MSGOUT	;PRINT IT
E804	CD 59 E7	1458	CALL	TAPWT	;WAIT FOR NULLS
E807	EB	1459	LOAD7A:	EX	DE,HL ;FLIP 'EM
E808	FD 5E 5E	1460	LD	E,(IY+THEAD+HSIZE);GET BLK	
E80B	FD 56 5F	1461	LD	D,(IY+THEAD+HSIZE+1)	
E80E	CD A9 E6	1462	LOAD8:	CALL	BLKADJ ;ADJUST BLOCK
E811	28 0F	1463	JR	Z,LOAD10	;THRU
E813	CD DA E2	1464	LOAD9:	CALL	TAPEIN ;GET BYTE
E816	CA D4 E1	1465	JP	Z,FINISH	;USER WANTS US
E819	77	1466	LD	(HL),A	;PUT AWAY
E81A	23	1467	INC	HL	;NEXT
E81B	10 F6	1468	DJNZ	LOAD9	;DO ALL BLOCKS
E81D	CD 4E E7	1469	CALL	CKCRC	;CHECK CRC
E820	18 EE	1470	JR	LOAD8	;LOOP FOR ALL BLOCKS
E822	CD AF E2	1471	LOAD10:	CALL	MTROFF ;SHUT UP
E825	21 53 E4	1472	LD	HL,FILHD	;POINT TO HEADING
E828	CD BA E1	1473	CALL	MSGOUT	;PRINT IT
E82B	CD DE E6	1474	CALL	HEDPRT	;GO PRINT HEADER
E82E	F1	1475	POP	AF	
E82F	F1	1476	POP	AF	
E830	F1	1477	POP	AF	
E831	C0	1478	RET	NZ	;WE'RE THRU
E832	FD 7E 5D	1479	LD	A,(IY+THEAD+HTYPE);GET FILE TYPE	
E835	E6 80	1480	AND	80H	;DATA FILE?
E837	C0	1481	RET	NZ	;YES! GO BACK
E838	FD 6E 62	1482	LD	L,(IY+THEAD+HXEQ);GET XEQ ADDR	
E83B	FD 66 63	1483	LD	H,(IY+THEAD+HXEQ+1)	
E83E	E9	1484	JP	(HL)	;GO DO IT!!
E83F	CD 34 E7	1485	LOADSK:	CALL	SKIFFL ;GO OVER IT!
E842	C3 BE E7	1486	JP	LOAD3A	

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

```

E845      1488 ;
E845      1489 ;
E845      1490 ;
E845      1491 ;      CHANGE PROMPT CHARACTER COMMAND
E845      1492 ;
E845      1493 ;
E845      1494 ;
E845      1495 PRMPTC: EQU    $
E845      1496      PUSH    IY
E847      E1      1497      POP     HL
E848      7E      1498 PRMP1: LD      A,(HL)
E849      FE 0D    1499      CP      CR
E84B      CA DE E1 1500      JP      Z,ERRPAR
E84E      FE 3D    1501      CP      '='
E850      23      1502      INC     HL
E851      20 F5    1503      JR      NZ,PRMP1
E853      7E      1504      LD      A,(HL)
E854      FD 77 44 1505      LD      (IY+PROMPT),A
E857      C9      1506      RET
E858      1507 ;
E858      1508 ;-----
E858      1509 ;
E858      1510 ;
E858      1511 ;
E858      1512 ;      BATCH COMMAND
E858      1513 ;
E858      1514 ;
E858      1515 BATCH: EQU    $
E858      FD 70 43 1516      LD      (IY+BATCHF),B;SET FLAG
E85B      C9      1517      RET

```



```

E85C      1519 ;
E85C      1520 ;
E85C      1521 ;
E85C      1522 ;      CREAT BATCH FILE COMMAND
E85C      1523 ;
E85C      1524 ;
E85C      1525 ;
E85C      E85C      1526 CREAT: EQU $
E85C      3E 2A      1527      LD A,'*'
E85E      CD 45 E0    1528      CALL CHROUT
E861      CD 3A E1    1529      CALL LINEIN ;GET A LINE
E864      FD E5      1530      PUSH IY ;MOVE IY
E866      E1         1531      POP HL ; TO HL
E867      7E         1532      LD A,(HL)
E868      FE 0D      1533      CP CR ;SEE IF END
E86A      C8         1534      RET Z ;YES!
E86B      06 01      1535      LD B,1
E86D      CD 8A E2    1536      CALL MOTRON ;TURN ON
E870      CD C2 E2    1537      CALL NULL ;SEND NULLS
E873      7E         1538 CREAT1: LD A,(HL) ;GET
E874      23         1539      INC HL ;NEXT
E875      CD EE E2    1540      CALL TAPOUT ;SEND
E878      FE 0D      1541      CP CR ;END?
E87A      20 F7      1542      JR NZ,CREAT1 ;NO
E87C      CD 9B E8    1543      CALL WRCRC ;WRITE CRC
E87F      CD AF E2    1544      CALL MTROFF ;OFF
E882      18 D8      1545      JR CREAT ;CONTINUE
E884      1546 ;
E884      1547 ;-----
E884      1548 ;
E884      1549 ;
E884      1550 ;      LIST BATCH FILE COMMAND
E884      1551 ;
E884      1552 ;
E884      1553 ;
E884      E884      1554 LIST: EQU $
E884      06 01      1555      LD B,1
E886      CD 8A E2    1556      CALL MOTRON ;TURN ON
E889      CD 05 E2    1557 LIST1: CALL CRLF ;NEW LINE
E88C      CD 59 E7    1558      CALL TAPWT ;WAIT FOR NULLS
E88F      CD DA E2    1559 LIST3: CALL TAPEIN ;GET
E892      FE 0D      1560      CP CR ;IS IT?
E894      28 F3      1561      JR Z,LIST1 ;YES!
E896      CD 45 E0    1562      CALL CHROUT ;PRINT IT
E899      18 F4      1563      JR LIST3 ;CONTINUE
E89B      1564 ;
E89B      1565 ;      WRITE CRC TO TAPE
E89B      1566 ;
E89B      E89B      1567 WRCRC: EQU $
E89B      FD 7E 46    1568      LD A,(IY+CRCBYT)
E89E      C3 EE E2    1569      JP TAPOUT ;GO WRITE & RETURN

```

```

E8A1      1571 ;
E8A1      1572 ;
E8A1      1573 ;
E8A1      1574 ;      MEMORY TEST COMMAND
E8A1      1575 ;
E8A1      1576 ;
E8A1      1577 ;
E8A1      E8A1      1578 TEST: EQU $
E8A1      CD 2F E2      1579      CALL SCANHL ;SKIP :TE"
E8A4      CA DE E1      1580      JP Z,ERRPAR
E8A7      CD 3D E2      1581      CALL CONV ;GET FROM
E8AA      D5      1582      PUSH DE ;AND SAVE IT
E8AB      CD 25 E2      1583      CALL SCAN ;SKIP DELIMS
E8AE      CA DE E1      1584      JP Z,ERRPAR
E8B1      CD 3D E2      1585      CALL CONV ;GET TO
E8B4      D5      1586      PUSH DE ;SAVE TO
E8B5      CD 25 E2      1587      CALL SCAN ;SEE IF CONTINUOUS
E8B8      D1      1588      POP DE
E8B9      E1      1589      POP HL ;GET FROM STACK
E8BA      01 01 00      1590      LD BC,1 ;SET PASS COUNTER
E8BD      FE 43      1591 TEST1: CP 'C' ;SET CONTINUOUS FLAGS
E8BF      F5      1592      PUSH AF ; IN STACK
E8C0      D5      1593      PUSH DE ;PUT BACK TO & FROM
E8C1      E5      1594      PUSH HL
E8C2      C5      1595      PUSH BC
E8C3      06 00      1596      LD B,0 ;CREAT MASK
E8C5      08      1597      EX AF,AF'
E8C6      AF      1598      XOR A
E8C7      08      1599      EX AF,AF'
E8C8      CD 2F E9      1600 TEST2: CALL REGRST
E8CB      70      1601 STUFF1: LD (HL),B ;PUT MASK IN MEM
E8CC      23      1602      INC HL ;NEXT MEMORY
E8CD      CD 3C E9      1603      CALL ENDCK ;SEE IF THRU
E8D0      20 F9      1604      JR NZ,STUFF1 ;GO ON IF NOT!
E8D2      CD D1 EA      1605      CALL QUIK ;SEE IF USER WANTS US
E8D5      CD 81 E9      1606      CALL STARPT ;PRINT "*"
E8D8      C2 D4 E1      1607      JP NZ,FINISH ;GO TO HIM IF SO
E8DB      CD 2F E9      1608      CALL REGRST ;GET TO & FROM
E8DE      78      1609 CHECK1: LD A,B
E8DF      BE      1610      CP (HL) ;IS IT OK?
E8E0      C4 42 E9      1611      CALL NZ,BADBYT ;NO-GO SAY SO!
E8E3      23      1612      INC HL ;NEXT ONE
E8E4      CD 3C E9      1613      CALL ENDCK ;END?
E8E7      20 F5      1614      JR NZ,CHECK1 ;NO!
E8E9      CD 85 E9      1615      CALL STARP2 ;ERASE "*"
E8EC      04      1616      INC B ;NEW MASK
E8ED      20 D9      1617      JR NZ,TEST2 ;CONTINUE FOR 255
E8EF      0E 00      1618      LD C,0 ;CREATE MASK
E8F1      41      1619 TEST3: LD B,C ;PUT IN PROPER PLACE
E8F2      CD 2F E9      1620      CALL REGRST
E8F5      70      1621 STUFF3: LD (HL),B
E8F6      23      1622      INC HL ;NEXT
E8F7      04      1623      INC B ;SHIFT MASK
E8F8      CD 3C E9      1624      CALL ENDCK ;END?
E8FB      20 F8      1625      JR NZ,STUFF3 ;NO
E8FD      41      1626      LD B,C ;RESET
E8FE      CD D1 EA      1627      CALL QUIK ;IS HE THERE?
E901      CD 81 E9      1628      CALL STARPT ;PRINT "*"

```

```

E904 C2 D4 E1      1629      JP      NZ,FINISH ;YES-GO TO HIM!
E907 CD 2F E9      1630      CALL    REGRST
E90A 78             1631 CHECK3: LD      A,B
E90B BE             1632      CP      (HL)      ;IS IT OK?
E90C C4 42 E9      1633      CALL    NZ,BADBYT ;NO!
E90F 23             1634      INC     HL      ;NEXT
E910 04             1635      INC     B      ;MASK TOO
E911 CD 3C E9      1636      CALL    ENDCK   ;THRU?
E914 20 F4          1637      JR      NZ,CHECK3 ;NO
E916 CD 85 E9      1638      CALL    STARP2   ;ERASE "*"
E919 0C             1639      INC     C
E91A 20 D5          1640      JR      NZ,TEST3
E91C             1641 ;
E91C D1             1642      POP     DE      ;GET PASS COUNT
E91D D5             1643      PUSH    DE      ;NOW-MOVE IT TO BC
E91E C1             1644      POP     BC
E91F CD 0F E2      1645      CALL    ADDCOL   ;PRINT PASS #
E922 21 AB E4      1646      LD      HL,PSCMSG ;PRINT PASS MESSAGE
E925 CD BA E1      1647      CALL    MSGOUT
E928 E1             1648      POP     HL      ;GET EVERYTHING OFF ST
E929 D1             1649      POP     DE
E92A F1             1650      POP     AF
E92B C0             1651      RET     NZ      ;NOT CONTINUOUS
E92C 03             1652      INC     BC      ;NEW PASS
E92D 18 8E          1653      JR      TEST1   ;GO START OVER
E92F             1654 ;
E92F             1655 ; TEST PROGRAM UTILITIES
E92F             1656 ;
E92F             1657 ;
E92F E92F          1658 REGRST: EQU    $
E92F D9             1659      EXX
E930 E1             1660      POP     HL      ;SAVE REGISTERS.
E931 D1             1661      POP     DE      ;GET RET ADDRESS
E932 D9             1662      EXX      ;TWICE
E933 E1             1663      POP     HL      ;POINT OUR REGS
E934 D1             1664      POP     DE
E935 D5             1665      PUSH    DE      ;PUT BACK
E936 E5             1666      PUSH    HL
E937 D9             1667      EXX
E938 D5             1668      PUSH    DE      ;PUT BACK RETURNS
E939 E5             1669      PUSH    HL
E93A D9             1670      EXX
E93B C9             1671      RET
E93C             1672 ;
E93C E93C          1673 ENDCK: EQU    $
E93C 7A             1674      LD      A,D
E93D BC             1675      CP      H
E93E C0             1676      RET     NZ
E93F 7B             1677      LD      A,E
E940 BD             1678      CP      L
E941 C9             1679      RET
E942             1680 ;
E942 E942          1681 BADBYT: EQU    $
E942 08             1682      EX      AF,AF'   ;GET FHEADING FLAGS
E943 20 0B          1683      JR      NZ,BADB2   ;GO AROUND HEADING
E945 E5             1684      PUSH    HL      ;WE NEED
E946 21 75 E4      1685      LD      HL,TESTHD ;POINT HEADING
E949 CD BA E1      1686      CALL    MSGOUT   ;PRINT IT

```

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

E94C	E1	1687	POP	HL	
E94D	3E 55	1688	LD	A,55H	;SET HEADING FLAGS
E94F	B7	1689	OR	A	;SET THOSE FLAGS!
E950	08	1690	BADB2: EX	AF,AF'	;PUT IT BACK!
E951	C5	1691	PUSH	BC	
E952	D5	1692	PUSH	DE	;I NEED DE
E953	EB	1693	EX	DE,HL	; FOR HL
E954	CD D1 EA	1694	CALL	QUICK	;SEE IF HE'S THERE
E957	C2 D4 E1	1695	JP	NZ,FINISH	;YEP-GO TO HIM
E95A	CD 0F E2	1696	CALL	ADDCOL	;PRINT ADDRESS
E95D	06 05	1697	LD	B,5	;PRINT 5 SPACES
E95F	CD D2 E2	1698	CALL	SPACES	
E962	EB	1699	EX	DE,HL	
E963	D1	1700	POP	DE	;RESTORE EVERYTHING
E964	0E 01	1701	LD	C,1	;CREATE MASK
E966	7E	1702	BADBY2: LD	A,(HL)	
E967	A8	1703	XOR	B	;SET ERROR BITS
E968	A1	1704	AND	C	;PEEL OFF CURRENT BITS
E969	E5	1705	PUSH	HL	;I NEED
E96A	20 10	1706	JR	NZ,BADBIT	
E96C	21 A6 E4	1707	LD	HL,OKMSG	
E96F	CD BA E1	1708	BADBY3: CALL	MSGOUT	;PRINT MESSAGE
E972	E1	1709	POP	HL	;RESTORE
E973	CB 21	1710	SLA	C	;ROTATE MASK
E975	30 EF	1711	JR	NC,BADBY2	;LOOP TILL ROTATE THRU
E977	C1	1712	POP	BC	;RESTORE 'C'
E978	CD 05 E2	1713	CALL	CRLF	;NEW LINE!
E97B	C9	1714	RET		
E97C	21 A1 E4	1715	BADBIT: LD	HL,BADMSG	;POINT BAD MSG
E97F	18 EE	1716	JR	BADBY3	;PRINT IT UP THERE!
E981		1717	STARPT: EQU	\$	
E981	3E 2A	1718	LD	A,'*'	
E983	18 02	1719	JR	STARP3	;GO PRINT IT
E985	3E 08	1720	STARP2: LD	A,CNTRLH	;FORM BACKSPACE
E987	C3 0C E0	1721	STARP3: JP	SEND	;GO PRINT IT & RET

```

E98A      1723 ;
E98A      1724 ;
E98A      1725 ;
E98A      1726 ;      PROM PACK COMMAND
E98A      1727 ;
E98A      1728 ;
E98A      1729 ;
E98A      E98A      1730 PROMPK: EQU      $
E98A      CD 2F E2      1731      CALL      SCANHL      ;SKIP "PP"
E98D      C2 FD DF      1732 PROMP1: JP      NZ,PCOLD      ;COLD START
E990      C3 FA DF      1733      JP      PWARM      ;WARM START
E993      1734 ;
E993      1735 ;      PROM PACK EQUATES
E993      1736 ;
E993      DFFD      1737 PCOLD: EQU      0DFFDH
E993      DFFA      1738 PWARM: EQU      0DFFAH
E993      1739 ;
E993      1740 ;
E993      1741 ;
E993      1742 ;
E993      1743 ;
E993      1744 ;      CENTRONICS PRINTER DRIVER
E993      1745 ;
E993      1746 ;
E993      E993      1747 CENDRV: EQU      $
E993      1748 ;
E993      F5      1749      PUSH      AF
E994      CD 1B E0      1750      CALL      VIDEO
E997      FE 0A      1751      CP      LF
E999      28 14      1752      JR      Z,CENGBK
E99B      F5      1753      PUSH      AF
E99C      DB FF      1754 CENBSY: IN      A,OFFH
E99E      CB 7F      1755      BIT      7,A
E9A0      20 FA      1756      JR      NZ,CENBSY
E9A2      F1      1757      POP      AF
E9A3      F6 80      1758      OR      80H
E9A5      D3 FF      1759      OUT      OFFH,A
E9A7      E6 7F      1760      AND      7FH
E9A9      D3 FF      1761      OUT      OFFH,A
E9AB      F6 80      1762      OR      80H
E9AD      D3 FF      1763      OUT      OFFH,A
E9AF      F1      1764 CENGBK: POP      AF
E9B0      C9      1765      RET

```

```

E9B1      1767 ;
E9B1      1768 ;
E9B1      1769 ;
E9B1      1770 ; VIDEO DRIVER ROUTINES
E9B1      1771 ;
E9B1      1772 ;
E9B1      1773 ;
E9B1      1774 ; INITIALIZE VIDEO BOARD
E9B1      1775 ;
E9B1      1776 VIDINT: EQU $
E9B1 21 80 F0 1777 LD HL,VID
E9B4 3E F8 1778 LD A,TOPHRG
E9B6 36 20 1779 CLR1: LD (HL),SPACE
E9B8 23 1780 INC HL
E9B9 BC 1781 CP H
E9BA 20 FA 1782 JR NZ,CLR1
E9BC 65 1783 LD H,L
E9BD FD 75 68 1784 LD (IY+LINE),L
E9C0 FD 74 69 1785 LD (IY+LINE+1),H
E9C3 FD 75 6A 1786 LD (IY+CHR),L
E9C6 FD 74 6B 1787 LD (IY+CHR+1),H
E9C9 CD 10 EB 1788 CALL WCSET ;MOVE USER CHR SET
E9CC 1789 ;
E9CC 1790 ; WRITE CURSOR ROUTINE
E9CC 1791 ;
E9CC 1792 WCUR: EQU $
E9CC CD D6 E9 1793 CALL PTRSET
E9CF 7E 1794 LD A,(HL)
E9D0 FD 77 67 1795 LD (IY+VDHLD),A
E9D3 36 5F 1796 LD (HL),05FH
E9D5 C9 1797 RET
E9D6 1798 ;
E9D6 1799 ; SET CURSOR POINTER ROUTINE
E9D6 1800 ;
E9D6 1801 PTRSET: EQU $
E9D6 21 80 F0 1802 LD HL,VID
E9D9 FD 5E 68 1803 LD E,(IY+LINE)
E9DC FD 56 69 1804 LD D,(IY+LINE+1)
E9DF 19 1805 ADD HL,DE
E9E0 FD 5E 6A 1806 LD E,(IY+CHR)
E9E3 FD 56 6B 1807 LD D,(IY+CHR+1)
E9E6 19 1808 ADD HL,DE
E9E7 C9 1809 RET
E9E8 1810 ;
E9E8 1811 ; REPLACE CHARACTER UNDER CURSOR
E9E8 1812 ;
E9E8 1813 REC: EQU $
E9E8 CD D6 E9 1814 CALL PTRSET
E9EB FD 7E 67 1815 LD A,(IY+VDHLD)
E9EE 77 1816 LD (HL),A
E9EF C9 1817 RET

```

```

E9F0          1819 ;
E9F0          1820 ;
E9F0          1821 ;
E9F0          1822 ;      VIDEO DRIVER ENTRY POINT
E9F0          1823 ;
E9F0          1824 ;
E9F0          1825 ;
E9F0          E9F0      1826 CHROT1: EQU $
E9F0          FD E5      1827          PUSH IY
E9F2          CD A2 E1    1828          CALL GETIY
E9F5          F5          1829          PUSH AF
E9F6          C5          1830          PUSH BC
E9F7          D5          1831          PUSH DE
E9F8          E5          1832          PUSH HL
E9F9          4F          1833          LD C,A
E9FA          CD E8 E9    1834          CALL REC      ;REPLACE CURSOR
E9FD          79          1835          LD A,C
E9FE          FE 0C       1836          CP OCH      ;FORM FEED
EA00          28 43       1837          JR Z,FRMFED
EA02          FE 0D       1838          CP CR      ;CAR RET
EA04          28 44       1839          JR Z,CARRET
EA06          FE 0A       1840          CP LF      ;LINE FEED
EA08          28 45       1841          JR Z,LINFED
EA0A          FE 17       1842          CP CNTRLW
EA0C          CA A3 EA    1843          JP Z,CURUP
EA0F          FE 1A       1844          CP CNTRLZ
EA11          28 3C       1845          JR Z,LINFED
EA13          FE 01       1846          CP CNTRLA
EA15          CA BA EA    1847          JP Z,CURLFT
EA18          FE 13       1848          CP CNTRLS
EA1A          28 18       1849          JR Z,CURRGT
EA1C          FE 08       1850          CP CNTRLH      ;BACKSPACE?
EA1E          28 72       1851          JR Z,BAKSPC
EA20          FE 11       1852          CP CNTRLQ      ;HOME?
EA22          CA C3 EA    1853          JP Z,HOMECU
EA25          FE 20       1854          CP SPACE
EA27          30 0A       1855          JR NC,OKDATA
EA29          CD CC E9    1856          CALL WCUR
EA2C          1857 ;
EA2C          EA2C      1858 RETURN: EQU $
EA2C          E1          1859          POP HL
EA2D          D1          1860          POP DE
EA2E          C1          1861          POP BC
EA2F          F1          1862          POP AF
EA30          FD E1       1863          POP IY
EA32          C9          1864          RET
EA33          1865 ;
EA33          1866 ;      DATA OK FOR DISPLAY
EA33          1867 ;
EA33          EA33      1868 OKDATA: EQU $
EA33          71          1869          LD (HL),C
EA33          EA34      1870 CURRGT: EQU $
EA34          13          1871          INC DE
EA35          7B          1872          LD A,E
EA36          E6 3F       1873          AND 3FH
EA38          28 06       1874          JR Z,NXLOC
EA3A          FD 73 6A     1875 OKDAT1: LD (IY+CHR),E
EA3D          FD 72 6B     1876          LD (IY+CHR+1),D

```

EA40	CD CC E9	1877	NXLOC:	CALL	WCUR
EA43	18 E7	1878		JR	RETURN
EA45		1879			
EA45		1880			FORM FEED
EA45		1881			
	EA45	1882	FRMFED:	EQU	\$
EA45	CD B1 E9	1883		CALL	VIDINT
EA48	18 E2	1884		JR	RETURN
A4A		1885			
EA4A		1886			CARRIAGE RETURN
EA4A		1887			
	EA4A	1888	CARRET:	EQU	\$
EA4A	11 00 00	1889		LD	DE,0
EA4D	18 EB	1890		JR	OKDAT1
EA4F		1891			
EA4F		1892			LINE FEED ROUTINE
EA4F		1893			
	EA4F	1894	LINFED:	EQU	\$
EA4F	FD 5E 68	1895		LD	E,(IY+LINE)
EA52	FD 56 69	1896		LD	D,(IY+LINE+1)
EA55	7B	1897		LD	A,E
EA56	E6 C0	1898		AND	0COH
EA58	CB 07	1899		RLC	A
EA5A	CB 07	1900		RLC	A
EA5C	CB 22	1901		SLA	D
EA5E	CB 22	1902		SLA	D
EA60	B2	1903		OR	D
EA61	FE 1D	1904		CP	1DH
EA63	28 15	1905		JR	Z,LLN
EA65	FD 6E 68	1906		LD	L,(IY+LINE)
EA68	FD 66 69	1907		LD	H,(IY+LINE+1)
EA6B	11 40 00	1908		LD	DE,64
EA6E	19	1909		ADD	HL,DE
A6F	FD 75 68	1910		LD	(IY+LINE),L
A72	FD 74 69	1911		LD	(IY+LINE+1),H
EA75	CD CC E9	1912		CALL	WCUR
EA78	18 B2	1913		JR	RETURN
EA7A	11 80 F0	1914	LLN:	LD	DE,VID
EA7D	21 C0 F0	1915		LD	HL,VID+64
EA80	01 40 07	1916		LD	BC,2048-192
EA83	ED B0	1917		LDIR	
EA85	3E BF	1918	LLN1:	LD	A,0BFH
EA87	36 20	1919		LD	(HL),SPACE
EA89	2B	1920		DEC	HL
EA8A	BD	1921		CP	L
EA8B	20 F8	1922		JR	NZ,LLN1
EA8D	CD CC E9	1923		CALL	WCUR
EA90	18 9A	1924		JR	RETURN
EA92		1925			
EA92		1926			BACKSPACE ROUTINE
EA92		1927			
	EA92	1928	BAKSPC:	EQU	\$
EA92	3E 01	1929		LD	A,CNTRLA
EA94	CD 45 E0	1930		CALL	CHROUT
EA97	3E 20	1931		LD	A,SPACE
EA99	CD 45 E0	1932		CALL	CHROUT
EA9C	3E 01	1933		LD	A,CNTRLA
EA9E	CD 45 E0	1934		CALL	CHROUT



EAA1	18 89	1935	JR	RETURN
EAA3		1936 ;		
EAA3		1937 ;	CURSOR UP ROUTINE	
EAA3		1938 ;		
	EAA3	1939	CURUP:	EQU \$
EAA3	FD 5E 68	1940	LD	E,(IY+LINE)
EAA6	FD 56 69	1941	LD	D,(IY+LINE+1)
EAA9	7B	1942	LD	A,E
EAAA	B2	1943	OR	D
EAAB	28 E3	1944	JR	Z,BAKSPC-2
EAAD	EB	1945	EX	DE,HL
EAAE	11 C0 FF	1946	LD	DE,-64
EAB1	19	1947	ADD	HL,DE
EAB2	FD 75 68	1948	LD	(IY+LINE),L
EAB5	FD 74 69	1949	LD	(IY+LINE+1),H
EAB8	18 86	1950	JR	NXLOC
EABA		1951 ;		
EABA		1952 ;	CURSOR LEFT ROUTINE	
EABA		1953 ;		
	EABA	1954	CURLFT:	EQU \$
EABA	7A	1955	LD	A,D
EABB	B3	1956	OR	E
EABC	CA 2C EA	1957	JP	Z,RETURN
EABF	1B	1958	DEC	DE
EAC0	C3 3A EA	1959	JP	OKDAT1
EAC3		1960 ;		
EAC3		1961 ;	HOME UP CURSOR ROUTINE	
EAC3		1962 ;		
	EAC3	1963	HOMEUC:	EQU \$
EAC3	FD 36 68 00	1964	LD	(IY+LINE),0
EAC7	FD 36 69 00	1965	LD	(IY+LINE+1),0;HOME UP LINE COUNT
EACB	11 00 00	1966	LD	DE,0
EACE	C3 3A EA	1967	JP	OKDAT1 ;GO HOME UP CHR AND RE

```

EAD1      1969 ;
EAD1      1970 ;
EAD1      1971 ;
EAD1      1972 ;      KEYBOARD QUICK CHECK
EAD1      1973 ;
EAD1      1974 ;      SCANS FOR :      CONTROL C
EAD1      1975 ;                      ESCAPE
EAD1      1976 ;                      RUN / STOP
EAD1      1977 ;
EAD1      1978 ;
      EAD1      1979 QUIK: EQU $
      EAD1      1980 ESCCHK: EQU $      ;FOR PREVIOUS ROUTINES
EAD1 FD E5      1981      PUSH IY
EAD3 CD A2 E1    1982      CALL GETIY
EAD6 FD 7E 45    1983      LD A,(IY+CMTRFG);GET MOTOR FLAG
EAD9 F6 01      1984      OR 1      ;SET MASK
EADB D3 FE      1985      OUT KYPORT,A
EADD DB FE      1986      IN A,KYPORT
EADF CB 67      1987      BIT 4,A
EAE1 28 13      1988      JR Z,QUIK1
EAE3 FD 7E 45    1989      LD A,(IY+CMTRFG);GET FLAGS
EAE6 E6 F0      1990      AND 0FH      ;MASK FOR ZERO
EAE8 D3 FE      1991      OUT KYPORT,A
EAEA DB FE      1992      IN A,KYPORT
EAEC CB 47      1993      BIT 0,A
EAE E 28 06      1994      JR Z,QUIK1
EAF0 CB 57      1995      BIT 2,A
EAF2 28 06      1996      JR Z,QUIK3
EAF4 18 11      1997      JR QUKRT1
EAF6 3E 1B      1998 QUIK1: LD A,ESC      ;FORM ESCAPE
EAF8 18 12      1999      JR QUKRET
EAFA FD 7E 45    2000 QUIK3: LD A,(IY+CMTRFG);GET MOTOR FLAGS
EAFD F6 03      2001      OR 3      ;SET MASK BITS
EAF F D3 FE      2002      OUT KYPORT,A
B01 DB FE      2003      IN A,KYPORT
EB03 CB 47      2004      BIT 0,A
EB05 28 03      2005      JR Z,QUIK4
EB07 AF      2006 QUKRT1: XOR A
EB08 18 02      2007      JR QUKRET
EBOA 3E 03      2008 QUIK4: LD A,CNTRLC
EBOC B7      2009 QUKRET: OR A
EB0D FD E1      2010      POP IY
EB0F C9      2011      RET

```

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

EB10		2013 ;	
EB10		2014 ;	
EB10		2015 ;	
EB10		2016 ;	WRITE USER CHARACTER SET
EB10		2017 ;	
EB10		2018 ;	
EB10		2019 ;	
	EB10	2020	WCSET: EQU \$
EB10	21 FE ED	2021	LD HL,CHRSET
EB13	11 00 FC	2022	LD DE,0FC00H
EB16	01 00 02	2023	LD BC,512
EB19	ED B0	2024	LDIR
EB1B	C9	2025	RET

```

EB1C      2027 ;
EB1C      2028 ;
EB1C      2029 ;
EB1C      2030 ;      KEYBOARD INPUT ROUTINE
EB1C      2031 ;
EB1C      2032 ;
EB1C      2033 ;
EB1C      2034 CHRIN1: EQU $
EB1C      2035 ;*****
EB1C      2036 ;KEYBOARD INPUT ROUTINE
EB1C      2037 ;VERSION 1.0      4/17/78
EB1C      00FE      2038 KYPORT: EQU OFEH      ;ON EXIDY COMPUTER
EB1C      0100      2039 DTIME: EQU 100H      ;TO BE DETERMINED
EB1C      2040 ;*****
EB1C      2041 ;
EB1C      2042 ;
EB1C      2043 ;INITIALIZATION
EB1C      2044 ;
EB1C      FD E5      2045 KEYBD: PUSH IY
EB1E      CD A2 E1    2046 CALL GETIY
EB21      C5          2047 PUSH BC      ;PUSH REGS ON STACK
EB22      D5          2048 PUSH DE
EB23      E5          2049 PUSH HL
EB24      DD E5       2050 PUSH IX
EB26      3E 01       2051 LD A,1      ;LOOK FOR REPEAT KEY
EB28      D3 FE       2052 OUT KYPORT,A ;SEND MASK
EB2A      DB FE       2053 IN A,KYPORT
EB2C      CB 4F       2054 BIT 1,A      ;REPEAT?
EB2E      20 0E       2055 JR NZ,NORPT ;NO-GO ON
EB30      01 88 13    2056 LD BC,5000 ;DELAY FOR REPEAT
EB33      0B          2057 REPET: DEC BC
EB34      78          2058 LD A,B
EB35      B1          2059 OR C      ;DONE?
EB36      20 FB       2060 JR NZ,REPET ;NO-
EB38      FD 7E 6C    2061 LD A,(IY+LSTKEY);GET LAST KEY
EB3B      C3 10 EC    2062 JP FINEND ;GO BACK
EB3E      AF          2063 NORPT: XOR A ;CLEAR A
EB3F      0E FE       2064 LD C,KYPORT ;LOAD KEYBOARD PORT NO.
EB41      5F          2065 LD E,A ;CLEAR E
EB42      CB FB       2066 SET 7,E ;SET SCAN ONCE FLAG
EB44      16 00       2067 MLOOP: LD D,0 ;CLEAR SPECIAL KEY FLAGS
EB46      42          2068 LD B,D ;CLEAR NEW FLAGS REG
EB47      26 00       2069 LD H,0 ;CLEAR SECTION COUNTER
EB49      DD 21 1E EC 2070 LD IX,INSTBL ;LOAD INSTRUCTION TABLE POINTER
EB4D      ED 61       2071 SLOOP: OUT (C),H ;OUTPUT SECTION NO.
EB4F      2E 01       2072 LD L,1 ;LOAD BIT POSITION REG
EB51      ED 78       2073 BLOOP: IN A,(C) ;INPUT SECTION BYTE
EB53      A5          2074 AND L ;TEST BIT
EB54      C2 E2 EB    2075 JP NZ,ABIT1 ;JUMP IF BIT=1
EB57      E5          2076 PUSH HL ;LOAD DEBOUNCE TIMER
EB58      21 00 01    2077 LD HL,DTIME ;LOAD TIME
EB5B      2D          2078 DEBOUN: DEC L ;COUNT DOWN
EB5C      20 FD       2079 JR NZ,DEBOUN
EB5E      25          2080 DEC H
EB5F      20 FA       2081 JR NZ,DEBOUN
EB61      E1          2082 POP HL ;RESTORE HL
EB62      ED 78       2083 IN A,(C) ;INPUT SECTION BYTE
EB64      A5          2084 AND L ;TEST BIT

```

EB65	C2 E2 EB	2085	JP	NZ,ABIT1	;JUMP IF BIT=1
EB68	DD 56 00	2086	LD	D,(IX+0)	;LOAD SPECIAL KEY FLAGS
EB6B	CB 7A	2087	BIT	7,D	;TEST FOR CODED KEY
EB6D	28 1A	2088	JR	Z,CODED	;JUMP IF CODED KEY
EB6F	CB 62	2089	BIT	4,D	;TEST FOR GRAPHIC KEY
EB71	28 02	2090	JR	Z,NONGRA	;JUMP IF NOT GRAPHIC KEY
EB73	CB F0	2091	SET	6,B	;SET GRAPHIC FLAG
EB75	CB 5A	2092	NONGRA: BIT	3,D	;TEST FOR CONTROL KEY
EB77	28 02	2093	JR	Z,NONCON	;JUMP IF NOT CONTROL KEY
EB79	CB E8	2094	SET	5,B	;SET CONTROL FLAG
EB7B	CB 52	2095	NONCON: BIT	2,D	;TEST FOR SHIFT KEY
EB7D	28 02	2096	JR	Z,NSHSHI	;JUMP IF NOT SHIFT KEY
EB7F	CB E0	2097	SET	4,B	;SET SHIFT FLAG
EB81	CB 4A	2098	NSHSHI: BIT	1,D	;TEST FOR SHIFT/LOCK KEY
EB83	28 5D	2099	JR	Z,ABIT1	;JUMP IF NOT SHIFT/LOCK KE
EB85	CB D8	2100	SET	3,B	;SET SHIFT/LOCK FLAG
EB87	18 59	2101	JR	ABIT1	
EB89	E5	2102	CODED: PUSH	HL	;CALCULATE TABLE POSITION
EB8A	D5	2103	PUSH	DE	
EB8B	DD E5	2104	PUSH	IX	
EB8D	E1	2105	POP	HL	
EB8E	11 1E EC	2106	LD	DE,INSTBL	
EB91	B7	2107	OR	A	;CLEAR CARRY
EB92	ED 52	2108	SBC	HL,DE	
EB94		2109	;DECIDE WHICH TABLE TO USE		
EB94	D1	2110	POP	DE	
EB95	CB 70	2111	BIT	6,B	;TEST FOR GRAPHIC KEY
EB97	28 15	2112	JR	Z,NOGRAP	;JUMP IF NO GRAPHIC KEY
EB99	CB 72	2113	BIT	6,D	;TEST FOR NONGRAPHIC CHAR.
EB9B	28 11	2114	JR	Z,NOGRAP	;JUMP IF NOT GRAPHIC CHAR.
EB9D	D5	2115	PUSH	DE	;CALCULATE TABLE POINTER
EB9E	11 6E EC	2116	LD	DE,GRATBL	;LOAD GRAPHIC TABLE START
EBA1	19	2117	ADD	HL,DE	
EBA2	7E	2118	LD	A,(HL)	;LOAD A WITH CODE
EBA3	CB FF	2119	SET	7,A	;SET GRAPHIC BIT
EBA5	D1	2120	POP	DE	;TEST FOR SHIFT
EBA6	CB 60	2121	BIT	4,B	
EBA8	28 26	2122	JR	Z,FINOP	;JUMP IF NO SHIFT
EBAA	CB F7	2123	SET	6,A	;SET SHIFT BIT
EBAC	18 22	2124	JR	FINOP	;JUMP TO FINISH OP.
EBAE	D5	2125	NOGRAP: PUSH	DE	
EBAF	CB 68	2126	BIT	5,B	;TEST FOR CONTROL KEY
EBB1	28 05	2127	JR	Z,SKIP1	;SKIP IF NOT CONTROL KEY
EBB3	11 BE EC	2128	LD	DE,CONTRL	;LOAD CONTROL TABLE START
EBB6	18 15	2129	JR	SKIP4	
EBB8	CB 60	2130	SKIP1: BIT	4,B	;TEST FOR SHIFT KEY
EBBA	28 05	2131	JR	Z,SKIP2	;SKIP IF NOT SHIFT KEY
EBBC	11 0E ED	2132	LD	DE,SHITBL	;LOAD SHIFT TABLE START
EBBF	18 0C	2133	JR	SKIP4	
EBC1	CB 58	2134	SKIP2: BIT	3,B	;TEST FOR SHIFT/LOCK KEY
EBC3	28 05	2135	JR	Z,SKIP3	;SKIP IF NOT SHIFT/LOCK KE
EBC5	11 5E ED	2136	LD	DE,SLOTBL	;LOAD SHIFT/LOCK TABLE STAI
EBC8	18 03	2137	JR	SKIP4	
EBCA	11 AE ED	2138	SKIP3: LD	DE,UNSTBL	;LOAD UNSHIFT TABLE START
EBCD	19	2139	SKIP4: ADD	HL,DE	;SETUP POINTER
EBCE	D1	2140	POP	DE	
EBCF	7E	2141	LD	A,(HL)	;LOAD A WITH CODE
EBD0	CB E3	2142	FINOP: SET	4,E	;SET END OF SCAN FLAG

```

EBD2 CB DB          2143 SET 3,E      ;SET SECTION FLAG
EBD4 CB D3          2144 SET 2,E      ;SET BIT POSITION FLAG
EBD6 CB BB          2145 RES 7,E      ;RESET SCAN ONCE FLAG
EBD8 E1             2146 POP HL       ;WAIT FOR KEY TO BE RELEASED
EBD9 F5             2147 PUSH AF
EBDA ED 78          2148 WAITK: IN A,(C) ;INPUT SECTION BYTE
EBDC A5             2149 AND L
EBDD 28 FB          2150 JR Z,WAITK
EBDF F1             2151 POP AF
EBE0 18 0B          2152 JR BITEND
EBE2 CB 05          2153 ARIT1: RLC L      ;SHIFT L
EBE4 3E 20          2154 LD A,20H    ;TEST FOR LAST BIT POSITION
EBE6 BD             2155 CP L
EBE7 20 02          2156 JR NZ,SKIP5 ;SKIP IF NOT END
EBE9 CB D3          2157 SET 2,E      ;SET BIT POSITION FLAG
EBEB DD 23          2158 SKIP5: INC IX    ;INCREMENT TABLE POINTER
EBED CB 53          2159 BITEND: BIT 2,E  ;TEST FOR BIT END
EBEF CA 51 EB       2160 JP Z,BLOOP
EBF2 CB 93          2161 RES 2,E      ;RESET BIT FLAG
EBF4 CB 5B          2162 BIT 3,E      ;TEST FOR SECTION FLAG
EBF6 20 07          2163 JR NZ,SECEND ;JUMP TO SECTION END
EBF8 24             2164 INC H      ;INCREMENT SECTION
EBF9 3E 10          2165 LD A,16    ;TEST FOR END
EBFB BC             2166 CP H
EBFC C2 4D EB       2167 JP NZ,SLOOP ;STAY IN LOOP
EBFF CB 9B          2168 SECEND: RES 3,E    ;RESET SECTION FLAG
EC01 37             2169 SCF      ;SET CARRY
EC02 CB 7B          2170 BIT 7,E      ;TEST SCAN ONCE FLAG
EC04 28 03          2171 JR Z,SKIP6
EC06 AF             2172 XOR A      ;CLEAR A AND CARRY
EC07 CB E3          2173 SET 4,E      ;SET END OF SCAN
EC09 CB 63          2174 SKIP6: BIT 4,E    ;TEST FOR END OF SCAN
EC0B 20 03          2175 JR NZ,FINEND ;JUMP TO FINISH TEST
EC0D C3 44 EB       2176 JP MLOOP
EC10 DD E1          2177 FINEND: POP IX    ;RESTORE REGISTERS
EC12 E1             2178 POP HL
EC13 D1             2179 POP DE
EC14 C1             2180 POP BC
EC15 B7             2181 OR A
EC16 28 03          2182 JR Z,KEYRET ;NO CHAR TODAY
EC18 FD 77 6C       2183 LD (IY+LSTKEY),A;SAVE IN CASE REPEAT
EC1B FD E1          2184 KEYRET: POP IY
EC1D C9             2185 RET      ;RETURN FROM SUBROUTINE
EC1E               2186 ;
EC1E 00 90 88 82    2187 INSTBL: DB 0,90H,88H,82H,84H ;INSTRUCTION TABLE(0)
      84
EC23               2188 ;
EC23 40 80 00 40    2189 DB 40H,80H,0,40H,0 ;(1)
      00
EC28 40 40 40 40    2190 DB 40H,40H,40H,40H,40H ;(2)
      40
EC2D 40 40 40 40    2191 DB 40H,40H,40H,40H,40H ;(3)
      40
EC32 40 40 40 40    2192 DB 40H,40H,40H,40H,40H ;(4)
      40
EC37 40 40 40 40    2193 DB 40H,40H,40H,40H,40H ;(5)
      40
EC3C 40 40 40 40    2194 DB 40H,40H,40H,40H,40H ;(6)

```

EC41	40 40 40 40 40 40	2195	DB	40H,40H,40H,40H,40H ;(7)
EC46	40 40 40 40 40	2196	DB	40H,40H,40H,40H,40H ;(8)
EC4B	40 40 40 40 40	2197	DB	40H,40H,40H,40H,40H ;(9)
EC50	40 40 40 40 40	2198	DB	40H,40H,40H,40H,40H ;(A)
EC55	40 00 00 40 40	2199	DB	40H,0,0,40H,40H ;(B)
EC5A	40 40 40 40 00	2200	DB	40H,40H,40H,40H,0 ;(C)
EC5F	40 40 40 40 40	2201	DB	40H,40H,40H,40H,40H ;(D)
EC64	40 40 00 40 40	2202	DB	40H,40H,0,40H,40H ;(E)
EC69	00 00 00 40 40	2203	DB	0,0,0,40H,40H ;(F)
EC6E	00 00 00 00 00	2204	GRATBL: DB	0,0,0,0,0 ;GRAPHIC TABLE(0)
EC73	0C 00 00 0D 00	2205	DB	0CH,0,0,0DH,0 ;(1)
EC78	28 27 1A 0E 00	2206	DB	28H,27H,1AH,0EH,0 ;(2)
EC7D	29 1C 1B 0F 01	2207	DB	29H,1CH,1BH,0FH,1 ;(3)
EC82	1D 11 10 03 02	2208	DB	1DH,11H,10H,3,2 ;(4)
EC87	2B 2A 1E 12 04	2209	DB	2BH,2AH,1EH,12H,4 ;(5)
EC8C	2D 2C 1F 13 05	2210	DB	2DH,2CH,1FH,13H,5 ;(6)
EC91	21 15 20 14 06	2211	DB	21H,15H,20H,14H,6 ;(7)
EC96	2E 22 16 08 07	2212	DB	2EH,22H,16H,8,7 ;(8)
EC9B	30 2F 23 17 09	2213	DB	30H,2FH,23H,17H,9 ;(9)
ECA0	25 24 19 18 0A	2214	DB	25H,24H,19H,18H,0AH ;(A)
ECA5	26 00 00 0C 0B	2215	DB	26H,0,0,0CH,0BH ;(B)
ECAA	3C 38 35 31 00	2216	DB	3CH,38H,35H,31H,0 ;(C)
ECAF	3D 39 36 33 32	2217	DB	3DH,39H,36H,33H,32H ;(D)
ECB4	3E 3A 00 37 34	2218	DB	3EH,3AH,0,37H,34H ;(E)
ECB9	00 00 00 3F 3B	2219	DB	0,0,0,3FH,3BH ;(F)
ECBE	03 00 00 00 00	2220	CONTRL: DB	3,0,0,0,0 ;CONTROL TABLE(0)
ECC3	0C 00 20 0B 1B	2221	DB	0CH,0,20H,0BH,1BH ;(1)
ECC8	18 1A 01 11 31	2222	DB	18H,1AH,1,11H,31H ;(2)
ECCD	03 04 13 17	2223	DB	3,4,13H,17H,32H ;(3)

ECD2	32 06 12 05 34	2224	DB	6,12H,5,34H,33H ;(4)
ECD7	33 02 16 07 14	2225	DB	2,16H,7,14H,35H ;(5)
ECDC	35 0D 0E 08 19	2226	DB	0DH,0EH,8,19H,36H ;(6)
ECE1	36 0B 09 0A 15	2227	DB	0BH,9,0AH,15H,37H ;(7)
ECE6	37 2C 0C 0F 39	2228	DB	2CH,0CH,0FH,39H,38H ;(8)
ECEB	38 2F 2E 3B 10	2229	DB	2FH,2EH,3BH,10H,30H ;(9)
ECF0	30 1C 00 1D 1B	2230	DB	1CH,0,1DH,1BH,3AH ;(A)
ECF5	3A 1F 0D 0A 1E	2231	DB	1FH,0DH,0AH,1EH,2DH ;(B)
ECFA	2D 2B 2A 2F 2D	2232	DB	2BH,2AH,2FH,2DH,20H ;(C)
ECFF	20 30 31 01 17	2233	DB	30H,31H,01H,17H,37H ;(D)
ED04	37 2E 1A 11 13	2234	DB	2EH,1AH,11H,13H,39H ;(E)
ED09	39 00 00 00 3D	2235	DB	0,0,0,3DH,33H ;(F)
ED0E	33 03 00 00 00	2236	SHITBL: DB	3,0,0,0,0 ;SHIFT TABLE(0)
ED13	00 0C 00 20 09	2237	DB	0CH,0,20H,9,1BH ;(1)
ED18	1B 58 5A 41 51	2238	DB	58H,5AH,41H,51H,21H ;(2)
ED1D	21 43 44 53 57	2239	DB	43H,44H,53H,57H,22H ;(3)
ED22	22 46 52 45 24	2240	DB	46H,52H,45H,24H,23H ;(4)
ED27	23 42 56 47 54	2241	DB	42H,56H,47H,54H,25H ;(5)
ED2C	25 4D 4E 48 59	2242	DB	4DH,4EH,48H,59H,26H ;(6)
ED31	26 4B 49 4A 55	2243	DB	4BH,49H,4AH,55H,27H ;(7)
ED36	27 3C 4C 4F 29	2244	DB	3CH,4CH,4FH,29H,28H ;(8)
ED3B	28 3F 3E 2B 50	2245	DB	3FH,3EH,2BH,50H,30H ;(9)
ED40	30 7C 60 7D 7B	2246	DB	7CH,60H,7DH,7BH,2AH ;(A)
ED45	2A 7F 0D 0A 7E	2247	DB	7FH,0DH,0AH,7EH,3DH ;(B)
ED4A	3D 2B 2A 2F 2D	2248	DB	2BH,2AH,2FH,2DH,20H ;(C)
ED4F	20 30 31 01 17	2249	DB	30H,31H,01H,17H,37H ;(D)
ED54	37 2E 1A 11 13	2250	DB	2EH,1AH,11H,13H,39H ;(E)
ED59	39 00 00 00 3D	2251	DB	0,0,0,3DH,33H ;(F)
ED5E	33 1B 00 00 00	2252	SLOTBL: DB	1BH,0,0,0,0 ;SHIFT/LOCK TABLE(0)



ED63	00 0C 00 20 0B 1B	2253	DB	0CH,0,20H,0BH,1BH ;(1)
ED68	58 5A 41 51 31	2254	DB	58H,5AH,41H,51H,31H ;(2)
ED6D	43 44 53 57 32	2255	DB	43H,44H,53H,57H,32H ;(3)
ED72	46 52 45 34 33	2256	DB	46H,52H,45H,34H,33H ;(4)
ED77	42 56 47 54 35	2257	DB	42H,56H,47H,54H,35H ;(5)
ED7C	4D 4E 48 59 36	2258	DB	4DH,4EH,48H,59H,36H ;(6)
ED81	4B 49 4A 55 37	2259	DB	4BH,49H,4AH,55H,37H ;(7)
ED86	2C 4C 4F 39 38	2260	DB	2CH,4CH,4FH,39H,38H ;(8)
ED8B	2F 2E 3B 50 30	2261	DB	2FH,2EH,3BH,50H,30H ;(9)
ED90	5C 40 5D 5B 3A	2262	DB	5CH,40H,5DH,5BH,3AH ;(A)
ED95	5F 0D 0A 5E 2D	2263	DB	5FH,0DH,0AH,5EH,2DH ;(B)
ED9A	2B 2A 2F 2D 20	2264	DB	2BH,2AH,2FH,2DH,20H ;(C)
ED9F	30 31 34 38 37	2265	DB	30H,31H,34H,38H,37H ;(D)
EDA4	2E 32 35 36 39	2266	DB	2EH,32H,35H,36H,39H ;(E)
EDA9	00 00 00 3D 33	2267	DB	0,0,0,3DH,33H ;(F)
EDAE	1B 00 00 00 00	2268	UNSTBL: DB	1BH,0,0,0,0 ;UNSHIFTED TABLE(0)
EDB3	0C 00 20 0B 1B	2269	DB	0CH,0,20H,0BH,1BH ;(1)
EDB8	78 7A 61 71 31	2270	DB	78H,7AH,61H,71H,31H ;(2)
EDBD	63 64 73 77 32	2271	DB	63H,64H,73H,77H,32H ;(3)
EDC2	66 72 65 34 33	2272	DB	66H,72H,65H,34H,33H ;(4)
EDC7	62 76 67 74 35	2273	DB	62H,76H,67H,74H,35H ;(5)
EDCC	6D 6E 68 79 36	2274	DB	6DH,6EH,68H,79H,36H ;(6)
EDD1	6B 69 6A 75 37	2275	DB	6BH,69H,6AH,75H,37H ;(7)
EDD6	2C 6C 6F 39 38	2276	DB	2CH,6CH,6FH,39H,38H ;(8)
EDDB	2F 2E 3B 70 30	2277	DB	2FH,2EH,3BH,70H,30H ;(9)
EDE0	5C 40 5D 5B 3A	2278	DB	5CH,40H,5DH,5BH,3AH ;(A)
EDE5	5F 0D 0A 5E 2D	2279	DB	5FH,0DH,0AH,5EH,2DH ;(B)
EDEA	2B 2A 2F 2D 20	2280	DB	2BH,2AH,2FH,2DH,20H ;(C)
EDEF	30 31 34 38	2281	DB	30H,31H,34H,38H,37H ;(D)

EDF4	37 2E 32 35 36 39	2282	DB	2EH,32H,35H,36H,39H ;(E)
EDF9	00 00 00 3D 33	2283	DB	0,0,0,3DH,33H ;(F)

EDFE		2285 ;	
EDFE		2286 ;	
EDFE		2287 ;	
EDFE		2288 ;	USER DEFINABLE CHARACTER SET
EDFE		2289 ;	
EDFE		2290 ;	
EDFE		2291 ;	
	EDFE	2292 CHRSET: EQU	\$
EDFE	80 80 80 80	2293 DB	80H,80H,80H,80H,80H,80H,80H,80H
	80 80 80 80		
EE06	40 40 40 40	2294 DB	40H,40H,40H,40H,40H,40H,40H,40H
	40 40 40 40		
EE0E	20 20 20 20	2295 DB	20H,20H,20H,20H,20H,20H,20H,20H
	20 20 20 20		
EE16	10 10 10 10	2296 DB	10H,10H,10H,10H,10H,10H,10H,10H
	10 10 10 10		
EE1E	00 3C 7E FF	2297 DB	00H,3CH,7EH,0FFH,0FFH,7EH,3CH,00H
	FF 7E 3C 00		
EE26	04 04 04 04	2298 DB	4,4,4,4,4,4,4,4
	04 04 04 04		
EE2E	02 02 02 02	2299 DB	2,2,2,2,2,2,2,2
	02 02 02 02		
EE36	01 01 01 01	2300 DB	1,1,1,1,1,1,1,1
	01 01 01 01		
EE3E	00 3C 42 81	2301 DB	0,3CH,42H,81H,81H,42H,3CH,0
	81 42 3C 00		
EE46	FF 00 00 00	2302 DB	0FFH,0,0,0,0,0,0,0
	00 00 00 00		
EE4E	00 FF 00 00	2303 DB	0,0FFH,0,0,0,0,0,0
	00 00 00 00		
EE56	00 00 FF 00	2304 DB	0,0,0FFH,0,0,0,0,0
	00 00 00 00		
EE5E	00 00 00 FF	2305 DB	0,0,0,0FFH,0,0,0,0
	00 00 00 00		
EE66	00 00 00 71	2306 DB	0,0,0,71H,0BEH,24H,24H,24H
	BE 24 24 24		
EE6E	81 42 24 18	2307 DB	81H,42H,24H,18H,18H,24H,42H,81H
	18 24 42 81		
EE76	00 00 00 00	2308 DB	0,0,0,0,1,6,8,8
	01 06 08 08		
EE7E	00 00 00 00	2309 DB	0,0,0,0,0C0H,30H,8,8
	C0 30 08 08		
EE86	FF 80 80 80	2310 DB	0FFH,80H,80H,80H,80H,80H,80H,80H
	80 80 80 80		
EE8E	FF 01 01 01	2311 DB	0FFH,1,1,1,1,1,1,1
	01 01 01 01		
EE96	FF FE FC F8	2312 DB	0FFH,0FEH,0FCH,0F8H,0F0H,0E0H,0C0H,80H
	F0 E0 C0 80		
EE9E	FF 7F 3F 1F	2313 DB	0FFH,7FH,3FH,1FH,0FH,7,3,1
	0F 07 03 01		
EEA6	00 00 00 00	2314 DB	0,0,0,0,0FH,0FH,0FH,0FH
	0F 0F 0F 0F		
EEAE	00 00 00 00	2315 DB	0,0,0,0,0F0H,0F0H,0F0H,0F0H
	F0 F0 F0 F0		
EEB6	00 00 00 00	2316 DB	0,0,0,0,0FFH,0,0,0
	FF 00 00 00		
EEBE	10 38 7C FE	2317 DB	10H,38H,7CH,0FEH,0FEH,7CH,10H,38H
	FE 7C 10 38		

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

EEC6	00 66 FF FF	2318	DB	0,66H,0FFH,0FFH,7EH,3CH,1BH,0
	7E 3C 18 00			
EECE	08 08 08 06	2319	DB	8,8,8,6,1,0,0,0
	01 00 00 00			
EED6	08 08 08 30	2320	DB	8,8,8,30H,0C0H,0,0,0
	C0 00 00 00			
EEDE	80 80 80 80	2321	DB	80H,80H,80H,80H,80H,80H,80H,0FFH
	80 80 80 FF			
EEE6	01 01 01 01	2322	DB	1,1,1,1,1,1,1,0FFH
	01 01 01 FF			
EEEE	80 C0 E0 F0	2323	DB	80H,0C0H,0E0H,0F0H,0F8H,0FCH,0FEH,0FFH
	F8 FC FE FF			
EEF6	01 03 07 0F	2324	DB	1,3,7,0FH,1FH,3FH,7FH,0FFH
	1F 3F 7F FF			
EEFE	0F 0F 0F 0F	2325	DB	0FH,0FH,0FH,0FH,0,0,0,0
	00 00 00 00			
EF06	F0 F0 F0 F0	2326	DB	0F0H,0F0H,0F0H,0F0H,0,0,0,0
	00 00 00 00			
EF0E	08 08 08 08	2327	DB	8,8,8,8,8,8,8,8
	08 08 08 08			
EF16	18 3C 7E FF	2328	DB	18H,3CH,7EH,0FFH,7EH,3CH,1BH,0
	7E 3C 18 00			
EF1E	1C 1C 6B 7F	2329	DB	1CH,1CH,6BH,7FH,6BH,8,8,1CH
	6B 08 08 1C			
EF26	F0 F0 F0 F0	2330	DB	0F0H,0F0H,0F0H,0F0H,0FH,0FH,0FH,0FH
	0F 0F 0F 0F			
EF2E	0F 0F 0F 0F	2331	DB	0FH,0FH,0FH,0FH,0F0H,0F0H,0F0H,0F0H
	F0 F0 F0 F0			
EF36	F0 F0 F0 F0	2332	DB	0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
	F0 F0 F0 F0			
EF3E	0F 0F 0F 0F	2333	DB	0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH
	0F 0F 0F 0F			
EF46	FF FF FF FF	2334	DB	0FFH,0FFH,0FFH,0FFH,0,0,0,0
	00 00 00 00			
EF4E	00 00 00 00	2335	DB	0,0,0,0,0,0FFH,0FFH,0FFH,0FFH
	FF FF FF FF			
EF56	01 02 04 08	2336	DB	1,2,4,8,10H,20H,40H,80H
	10 20 40 80			
EF5E	80 40 20 10	2337	DB	80H,40H,20H,10H,8,4,2,1
	08 04 02 01			
EF66	08 08 08 08	2338	DB	8,8,8,8,0FFH,8,8,8
	FF 08 08 08			
EF6E	00 00 00 00	2339	DB	0,0,0,0,0,0,0FFH,0,0
	00 FF 00 00			
EF76	00 00 00 00	2340	DB	0,0,0,0,0,0,0,0FFH,0
	00 00 FF 00			
EF7E	00 00 00 00	2341	DB	0,0,0,0,0,0,0,0,0FFH
	00 00 00 FF			
EF86	55 AA 55 AA	2342	DB	55H,0AAH,55H,0AAH,55H,0AAH,55H,0AAH
	55 AA 55 AA			
EF8E	08 08 08 08	2343	DB	8,8,8,8,0FFH,0,0,0
	FF 00 00 00			
EF96	FF FF 00 00	2344	DB	0FFH,0FFH,0,0,0,0,0,0
	00 00 00 00			
EF9E	08 08 08 08	2345	DB	8,8,8,8,0FH,8,8,8
	0F 08 08 08			
EFA6	50 A0 50 A0	2346	DB	50H,0A0H,50H,0A0H,50H,0A0H,50H,0A0H
	50 A0 50 A0			

## EXIDY STANDARD MONITOR SOFTWARE

07/26/78

EFAE	C0 C0 C0 C0	2347	DB	0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
	C0 C0 C0 C0			
EFB6	03 03 03 03	2348	DB	3,3,3,3,3,3,3,3
	03 03 03 03			
EFBE	00 00 00 00	2349	DB	0,0,0,0,55H,0AAH,55H,0AAH
	55 AA 55 AA			
EFC6	08 08 08 08	2350	DB	8,8,8,8,0F8H,8,8,8
	F8 08 08 08			
EFCE	00 00 00 00	2351	DB	0,0,0,0,0,0,0FFH,0FFH
	00 00 FF FF			
EFD6	00 00 00 00	2352	DB	0,0,0,0,0FFH,8,8,8
	FF 08 08 08			
EFDE	00 00 00 00	2353	DB	0,0,0,0,0FH,8,8,8
	0F 08 08 08			
EFE6	00 00 00 00	2354	DB	0,0,0,0,0F8H,8,8,8
	F8 08 08 08			
EFEE	08 08 08 08	2355	DB	8,8,8,8,0FH,0,0,0
	0F 00 00 00			
EFF6	08 08 08 08	2356	DB	8,8,8,8,0F8H,0,0,0
	F8 00 00 00			

EXIDY STANDARD MONITOR SOFTWARE

07/26/78

```
EFFE      2358 ;  
EFFE      2359 ;  
EFFE      2360 ; -- END OF PROGRAM!!!!  
EFFE      2361 ;  
EFFE      2362 ;  
EFFE      2363      END
```

## CROSS REFERENCE

ABIT1	EBE2	2075	2085	2099	2101														
ADDCOL	E20F	0936	0960	0979	1645	1696													
ADDOUT	E1E8	0233	0239	0512	1281														
BADB2	E950	1683																	
BADBIT	E97C	1706																	
BADBY2	E966	1711																	
BADBY3	E96F	1716																	
BADBYT	E942	1611	1633																
BADMSG	E4A1	1715																	
BAKSPC	EA92	1851	1944																
BASLOD	E02D																		
BASSAV	E02A																		
BATCH	E858	0814																	
BATCHF	0043	0071	0319	0437	1516														
BITEND	EBED	2152																	
BKSPC	E175	0331																	
BLKADJ	E6A9	1206	1324	1462															
BLKAJ2	E6B6	1222																	
BLOOP	EB51	2160																	
BUFFER	0000	0065																	
CARRET	EA4A	1839																	
CENBSY	E99C	1756																	
CENDRV	E993	1135																	
CENGBK	E9AF	1752																	
CHEAD	0047	0075	0647	1112	1118	1119	1171	1172	1184	1185	1186	1198							
		1448																	
CHECK1	E8DE	1614																	
CHECK3	E90A	1637																	
CHR	006A	0084	1786	1787	1806	1807	1875	1876											
CHRIN	E030	0112	0322																
CHRIN1	EB1C <sup>25,235</sup>	0117	0216																
CHRINR	E041	0129	0149																
CHROT1	E9F0	0118	0219																
CHROUT	E045	0113	0261	0340	0351	0419	0482	0495	0497	0514	0516	0533							
		0729	0964	1265	1272	1283	1528	1562	1930	1932	1934								
CHRSET	EDFE	2021																	
CKCRC	E74E	0367	1309	1329	1469														
CLR1	E9B6	1782																	
CMOTOF	E027																		
CMOTON	E024																		
CMTRFG	0045	0073	0680	0700	1983	1989	2000												
CNTRLA	0001	1846	1929	1933															
CNTRLC	0003	2008																	
CNTRLH	0008	0350	1720	1850															
CNTRLQ	0011	1852																	
CNTRLS	0013	1848																	
CNTRLW	0017	1842																	
CNTRLZ	001A	1844																	
CODED	EB89	2088																	
COLD	E000																		
CONTRL	ECBE	2128																	
CONV	E23D	0919	0926	0978	0990	1007	1011	1016	1030	1046	1081	1169							
		1175	1191	1243	1413	1419	1581	1585											
CONV1	E240	0619																	
CONV2	E25B	0618																	
CR	000D	0326	0338	0365	0420	0494	0559	0572	0862	0864	0864	0866							
		0869	0869	0873	0876	0876	0883	0888	0888	0890	0890	0895							
		0895	0898	0902	0902	0905	0905	0907	1499	1533	1541	1560							

■

Code	Address	Value
CRCBYT	0046	1838
CRCMSG	E408	0074 0720 0765 0771 1219 1335 1361 1568
CRCOMP	E2FD	0445
CREAT	E85C	0747
CREAT1	E873	0806 1545
CRLF	E205	1542
CURLFT	EABA	0259 0334 0339 0916 0944 0955 0975 1285 1421 1557 1713
CURRGT	EA34	1847
CURUP	EEA3	1849
DEBOUN	EB5B	1843
DELAY	E2A0	2079 2081
DELAY1	E2A2	0695
DELAY2	E2A3	0689
DELAY3	E2A6	0688
DHEAD	E417	0929
DTIME	0100	2077
DUMP	E4D3	0794
DUMP1	E4EE	0959
DUMP2	E4F8	0956
DUMP3	E501	0954 0965
ENDCK	E93C	1603 1613 1624 1636
ENDTBL	E34E	
ENTER	E538	0796
ENTER1	E544	0986
ENTER2	E54F	0995
ERRCMD	E134	0266
ERRCRC	E1E3	1338
ERRMSG	E3DD	0433
ERRPAR	E1DE	0604 0608 0918 0977 1006 1010 1045 1059 1069 1075 1078
		1165 1166 1168 1174 1500 1580 1584
ESC	001B	1998
ESCCHK	EAD1	0741 0934
FILES	E6B9	0802
FILES1	E6C4	1242
FILES2	E6CD	1252 1254
FILES3	E6E4	1267
FILES4	E704	1284
FILHD	E453	1245 1472
FINEND	EC10	2062 2175
FINISH	E1D4	0362 0820 0935 1305 1327 1353 1358 1465 1607 1629 1695
FINOP	EBD0	2122 2124
FNDMSG	E4CA	1429
FRMFED	EA45	1837
GETHD1	E724	1308
GETHD2	E71B	1248 1424
GETIY	E1A2	0128 0141 0246 0673 0697 0740 0753 1828 1982 2046
GO	E597	0804
GRATBL	EC6E	2116
HADDR	0009	1171 1172 1437 1438
HCHOT2	E1FA	0475
HCHOT3	E200	0479
HCHOUT	E1ED	0466 0535
HEAD2	E3BC	0234
HEAD3	E3D5	0240
HEADLN	0010	1197 1303
HEDING	E362	0230
HEDFRT	E6DE	1249 1431 1474



### CROSS REFERENCE

[illegible]

•

MAIN4	E123	0273
MLOOP	EB44	2176
MOTR01	E297	0676
MOTRON	E28A	0121 0359 1193 1247 1422 1536 1556
MOVE	E562	0810
MOVE1	E589	1034
MOVE2	E58E	1015
MSG0T2	E1BE	0423
MSGOUT	E1BA	0231 0235 0241 0371 0421 0434 0436 0930 1246 1430 1457 1473 1647 1686 1708
MTROF1	E2B4	0438
MTROFF	E2AF	0122 0368 1207 1471 1544
NAMEN1	E27A	0659
NAMEN2	E282	0653
NAMFND	E264	1164 1402
NOGRAP	EBAE	2112 2114
NONCON	EB7B	2093
NONGRA	EB75	2090
NONSHI	EB81	2096
NORPT	EB3E	2055
NULL	E2C2	1194 1203 1537
NULL1	E2C4	0717
NUMBER	E255	0606
NXLOC	EA40	1874 1950
OKDAT1	EA3A	1890 1959 1967
OKDATA	EA33	1855
OKMSG	E4A6	1707
OUTADD	003F	0069 0151 0152 0220 0221 1136 1137
OUTAPE	E012	1132
OUTDLY	E051	0147
PARIN	E776 <sup>118, 231</sup>	0119 1374
PARLIN	E01E	1148
PARLOT	E021	1129
PAROT1	E780	1382
PAROUT	E77F	0120
PCOLD	DFFD	0222 1732
PRMP1	E848	1503
PRMPTC	E845	0818
PROMP1	E98D	0229
PROMPK	E98A	0822
PROMPT	0044	0072 0211 0260 1505
PSCMSG	E4AB	1646
PTRSET	E9D6	1793 1814
PWARM	DFFA	0203 0225 1733
QUIK	EAD1	0116 0202 1605 1627 1694
QUIK1	EAF6	1988 1994
QUIK3	EAFA	1996
QUIK4	EB0A	2005
QUIKCK	E015	
QUKRET	EB0C	1999 2007
QUKRT1	EB07	1997
RAM	0000	0179
RAMTOP	F000	0197 0198 0232 0387
REC	E9E8	1834
RECEVE	E009	
REGST	E92F	1600 1608 1620 1630
REPET	EB33	2060
RETURN	EA2C	1878 1884 1913 1924 1935 1957

**CROSS REFERENCE**

[illegible]